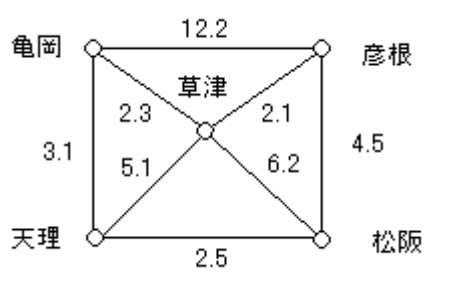
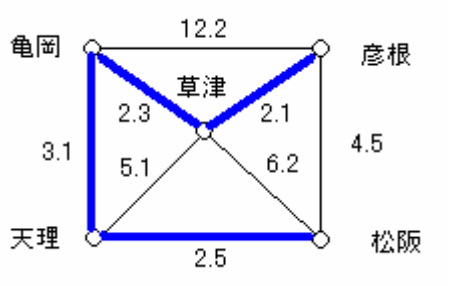
アルゴリズム論（担当　石井秀則）Ver.8.02

グラフの枝に重みをつけたものを重みつきグラフあるいはネットワークという。これは実社会や多くの科学分野で出てくる問題の数学的モデルとなっている。

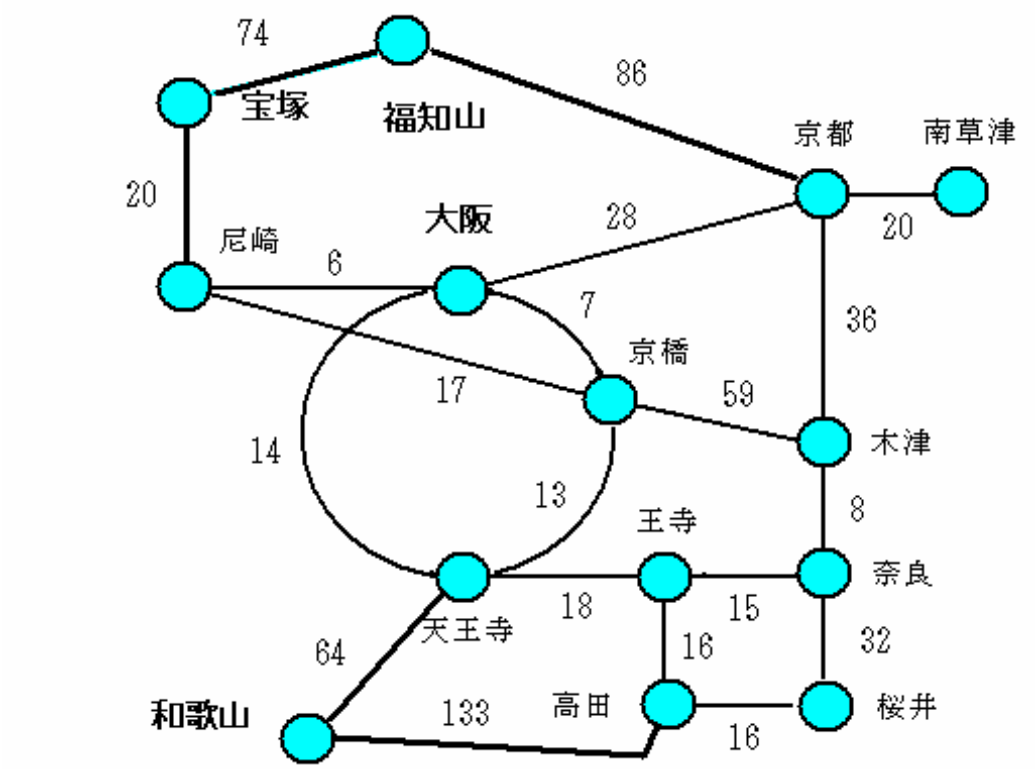
【最小木問題】

都市間の最小費用情報通信網構成問題を考えてみる。いくつかの都市の間に情報通信ケーブルを敷設する。各都市間の敷設費用の見積もりは業者から出ている。どの都市も（他の都市を経由してもよいから）必ず情報通信ケーブルで繋がるようにし、しかも総費用を最小にするには実際にどのように情報通信ケーブルを敷設すればよいかという問題である。例えば、草津、彦根、亀岡、天理、松阪の５都市間の例をあげると、次の図のようになる。枝の横の数字は２都市間の敷設費用見積もり額。これより、総費用を最小にする実際のプランは右図の太線のようになる。





【最短路問題】



この図はＪＲ西日本のいくつかの駅とその間の所要時間（分）である。例えば、南草津駅から和歌山へ行くにはいくつかの経路があるが、どの経路が一番速いか。電車に乗るのが好きな場合は、どの経路が一番遅いかという問題設定もありうる。

都市や駅の数が多くなると、このような問題を計算機で解決する必要が出来てくる。そのときに如何に早く解決を得るかというところが、アルゴリズムの腕のみせどころ。本講義の最終目的はこのようなネットワークに関わるアルゴリズムのいくつかを紹介することである。そのために、最初にグラフの基礎概念から始める。

【参考書】

- 小澤孝夫著「コンピュータ・アルゴリズム」昭晃堂
- ウイルソン著「グラフ理論入門」近代科学社
- ポリア他著「組み合わせ論入門」近代科学社
- オア著　「グラフ理論」河出書房新社
- ディーステル著「グラフ理論」Springer
- Chartrand 著「Introductory Graph Theory」 Dover
- Cormen 他著「Introduction to Algorithms」MIT press
- Foulds 著「Graph Theory Applications」 Springer-Verlag

１．グラフの定義と基本概念

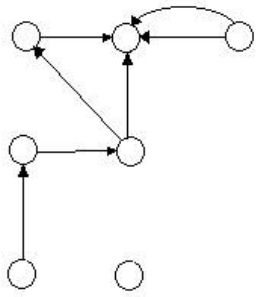
まず、グラフの定義から始める。

定義 1.1（最も一般な定義）：

V,E は集合で、∂は E から V×V への写像のとき、G=(V,E,∂)を有向グラフと言う。V の要素は節点（または頂点、vertex)、E の要素は枝（または辺、edge）と呼ばれる。枝 e に対し、∂(e)=(u,v)とおく。このとき u は e の始点、v は e の終点という。

通常、**V,E は有限個の要素から成ることを仮定する**のでここでもそれに従う。グラフは有限個の要素とその間の繋がりのみを表す概念と言える。

∂(e)=(u,v)のとき、u から v に向かって e で繋がっていると思うのである。節点を○で表し、枝を節点間を結ぶ矢印付きの線と思うと、次の図のようなものがグラフである。



この例の中にも見られるが、∂(e_1)=∂(e_2)となる $e_1 \neq e_2$ が存在してもかまわない。このような枝 e_1、e_2 を平行枝という。また、∂(e)=(u, u)となる枝は自己ループという。平行枝も自己ループも存在しないグラフは**単純グラフ**という。グラフの枝に向きを考えないとき、**無向グラフ**という。節点をいくつかの都市で枝をその都市間を結ぶ道路とみると、各道路で通行の向きが決まっているのが有向グラフで、どちら向きでも通行可能なときが無向グラフである。以下、∂は省略して、グラフ G=(V,E)とかく。グラフ G’=(V’,E’)が V’⊆V,E⊆E’をみたすとき、G’は G の**部分グラフ**という。無向グラフにおいて、節点 u と v を結ぶ枝があるとき、u と v は**隣接**しているという。また、有向グラフにおいては u を始点とし、v を終点とする枝があるとき、v は u に隣接しているという。

定義 1.2　無向グラフ G=(V,E)において、節点 v の次数 deg(v)は

deg(v)=v を端点とする枝の個数＝ v に隣接する節点の個数

と定義する。各枝には２つ端点があるので、

$\Sigma \deg(v)=2|E|$

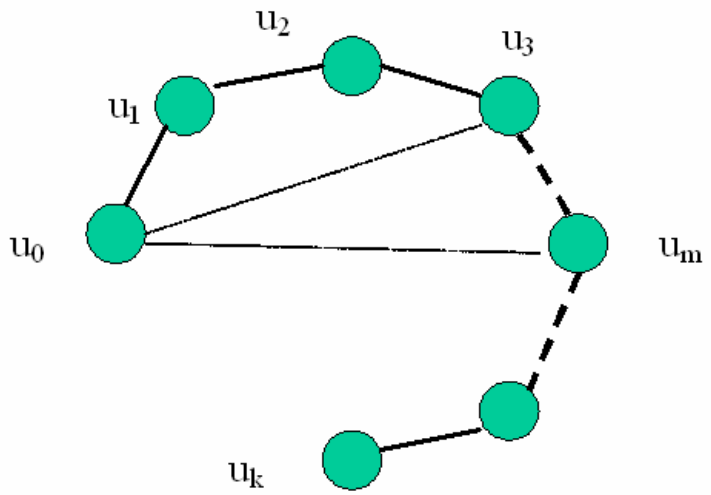
が成り立つ。ここで、和はすべての節点を走る。|E|は枝の総数。この式より、次数の総和は偶数であることがすぐにわかる。したがって、奇数次数の節点は存在しても偶数個である。

定義 1.3（道、閉路）

有向グラフG=(V,E)において、その枝の列ｐ：e_1、e_2、…、e_n　が　e_iの終点＝e_{i+1}の始点（i=1,2,…,n-1）をみたすとき、ｐを**道**（路、path）という。（ここで、枝e_iはすべて異なるとする。）道を構成する枝の個数をその**道の長さ**という。e_1 の始点をu，e_nの終点をｖとするとき、ｐはuからvへの道という。u＝vのとき、ｐは（有向）**閉路**という。無向グラフについても同様に道や閉路の概念が定義される。単純グラフについては節点u,vを結ぶ枝があれば唯１つなので、それを(u,v)と書く。道p:(u_0,u_1),(u_1,u_2),‥,(u_{k-1},u_k)　をp:$u_0,u_1,,u_{k-1},u_k$と書くこともある。$u_0,u_1,,u_{k-1},u_k$がすべて異なるとき、単純な道という。

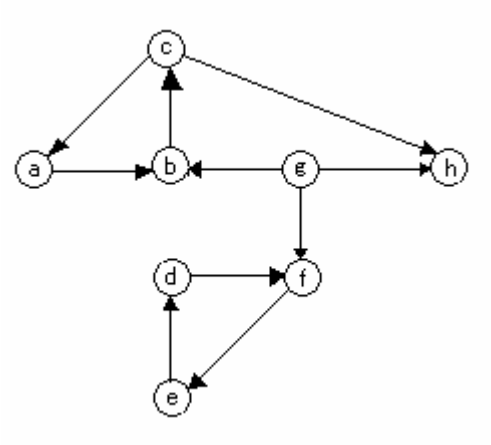
命題 1.4　　無向グラフ G において、任意の節点の次数が２以上ならば、閉路が存在する。その閉路の長さは G の最小次数より大きい。

証明　Gにおける単純で最長の道をｐ：$u_0u_1u_2\cdots u_k$ とする。このとき、ｐの最長性より、u_0に隣接する節点はすべてｐの上にある。つまり、$u_1u_2\cdots u_k$のどれかである。そのうち、添え字の最大なものをu_m とする。m≧deg(u_0)。このとき、ｃ：$u_0u_1u_2\cdots u_mu_0$　が求める閉路でその長さ＝ｍ＋１＞deg(u_0)≧Gの最小次数



定義 1.5（連結性）
無向グラフＧにおいて、任意の２節点 u,ｖに対して、ｕからｖへの道が存在するとき、Ｇは連結という。無向グラフＧにおいてｕからｖへの道が存在するとき、u～v とかくと、～はＶにおける同値関係である。（なお、ｕからｕへは０本の枝からなる道が存在すると考える。）この同値類で定まる部分グラフを Ｇの連結成分という。有向グラフについてはこれでは同値関係にならない。ｕからｖへの道とｖからｕへの道の両方が存在するとき、ｕとｖは強連結であるという。強連結という関係は同値関係で、この同値類で定まる部分グラフを Ｇの強連結成分という。

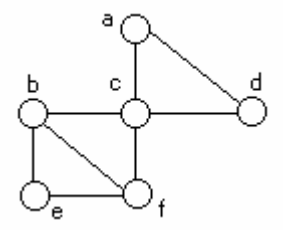
例 1.



この例で強連結成分は{a,b,c},{d,e,f},{g},{h}。このグラフの枝の向きを無視して無向グラフと見たとき、連結であることは明らかであろう。

連結無向グラフ G=(V,E)について、E の部分集合 F で、これを G から取り除くと、G が非連結となるようなものを非連結化集合という。C は非連結化集合で、C の任意の真の部分集合は非連結化集合でないとき、C を G のカットセットという。

例 2.　下図で C={(a,c),(c,d)}はカットセットである。{(a,c),(a,d)}もカットセット。



1 個の枝を取り除いたとき、非連結になるなら、この枝を橋(bridge)という。

G のカットセットの枝の個数の最小値を G の**枝連結度**といい、λ(G)と書く。つまり、G を非連結にするために除去すべき枝の最小個数のことである。上の例では橋はないので、

λ(G)=2 である。枝のかわりに節点を考えると、節点連結度の概念に到達する。すなわち、節点をいくつか除去する（このとき、節点につながっている枝もいっしょに除去する）と

非連結またはK_1（1 点からなる完全グラフ、後述）になるような節点の個数の最小値を**節点連結度**という。これをκ(G)と書く。上の例では節点cを除去すると、２つの連結成分に分かれるので、κ(G)=１である。枝連結度も節点連結度も重要な量であり、たとえば、グラフがコンピュータのネットワークを表すとき、どれだけのケーブルもしくはどれだけのコンピュータが故障すればこのネットワークがシステムとして機能しなくなるかということを判断することができる。

練習問題 1.1　ｎ≦5 を満たす各ｎについて、ｎ点からなる連結単純無向グラフをすべて求めよ。（互いに同型でないものを数えること）

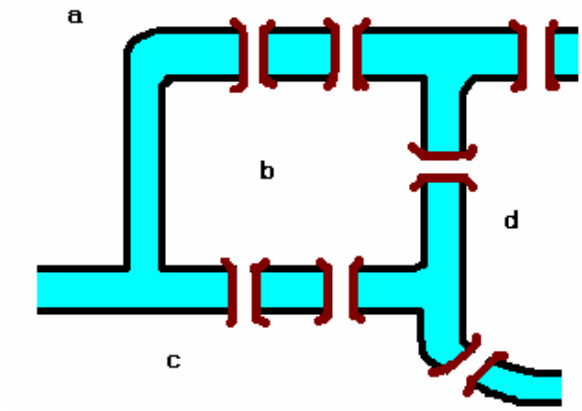
練習問題 1.2　有向グラフ G=(V,E)の各節点ｖについて、ｖを終点とする枝の個数を入次数　in-deg(v)、ｖを始点とする枝の個数を出次数　out-deg(v)という。$\Sigma_{v\in V}$ in-deg(v)、$\Sigma_{v\in V}$ out-deg(v)を求めよ。

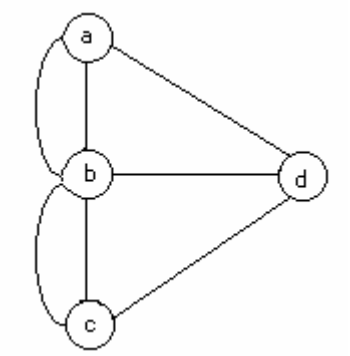
２．オイラーグラフ

さて、**無向グラフ**Ｇについて、すべての枝をちょうど１回含む閉路が存在するとき、その閉路を**オイラー閉路**という。オイラー閉路が存在するかどうかは、そのグラフがいわゆる一筆書きができるかということである。オイラー閉路が存在する無向グラフを**オイラーグラフ**という。グラフ理論の創始者はオイラー（1707-1783）で、グラフ理論に関する最初の論文(1736 年)は「ケーニヒスベルクの橋の問題」に端を発している。その問題を説明する前にオイラーについてすこし紹介したい。オイラー（Leonhard Euler）はスイスの Basel で 1707 年に生まれた。1726 年にペテルスブルクに移り、その後、ベルリンに移る（1741-66）が、再び、ペテルスブルクに戻り、その地で一生を終えている。

オイラーの業績は膨大で、数学の全分野に渡ると言っても過言でないくらい広範囲にわたる。オイラーの名を冠した数学的概念や公式・定理もいっぱいあって、例えば、整数論においてはオイラーの関数φ(n)やオイラーの基準、オイラー定数、そしてオイラー積、また、幾何学ではオイラー標数、解析学では変分問題と関連した偏微分方程式にオイラーの名前が残っている。

さて、「ケーニヒスベルクの橋の問題」というのは次のような問題である。ケーニヒスベルクという町にはクナイプホーフという島があり、それをとりまいてプレ―ゲル川の２つの支流が流れている。そして２つの支流には７つの橋がかかっている。問題というのはこれらの７つの橋をちょうど１回ずつ渡るような（最後は出発点に戻る）散歩道を計画できるかというものである。
そこで、オイラーは町は川によって、4 つの部分（図の a, b, c, d）に分かれているが、橋はその地域をつなぐものと考え、この問題の数学的モデルとして、４つの地域を節点、橋を枝として図のようなグラフを考え出した。





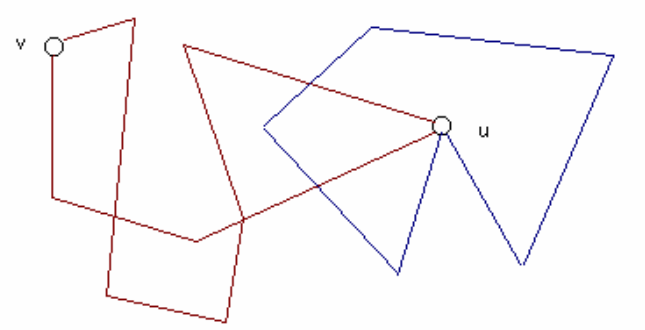
すると、当初の問題はこのグラフにおいて、すべての枝をちょうど１回ずつ通る閉路が存在するかということになる。つまり、今の言葉でいうとオイラー閉路が存在するかということになる。オイラーは一般的に連結無向グラフについて、オイラー閉路が存在するための必要十分条件を見つけた。それは非常

に判定しやすい明確な必要十分条件である。それを述べるために節点の次数という概念を導入する。すでに述べたように、無向グラフ G=(V,E)において、節点 v の次数 deg(v)は　　**deg(v)=v を端点とする枝の個数**
であった。次数の総和は偶数なので、奇数次数の節点は存在しても偶数個である。

定理 2.1（Euler）連結無向グラフ G=(V,E)において、オイラー閉路が存在するための必要十分条件はすべての節点の次数が偶数であることである。

上の「ケーニヒスベルクの橋の問題」の場合は deg(a)=deg(c)=deg(d)=3, deg(b)=5 なのでオイラー閉路は存在しない。つまり、7 つの橋をちょうど 1 回ずつ渡るような散歩道は残念ながら、ないのである。

（証明）必要性　オイラー閉路 p が存在するとする。 p に沿って歩き、節点 v を訪れるとき、 v に入れば、必ず v から出る。オイラー閉路であるから、同じ枝を通ることはないし、全ての枝を通る。したがって、deg(v)は偶数。
十分性　1 つの節点 v を固定し、 v から出発して、未だ通っていない枝をどんどん進み通っていない枝がなくなったところで終わる道 p を考える。すべての節点の次数が偶数なので、 v 以外の節点については入れば必ず出られる。したがって、 p の終点は最初の v である。つまり、 p は閉路。 p がすべての枝を含んでいればこれが求めるもの。そこで、 p に含まれる節点 u で、u を端点とする未だ通っていない枝があるとする。G は連結だから p が全体でなければ必ずこのような u が存在する。u において、{未だ通っていない枝で u を端点とするもの} の個数は偶数である。すべての節点についてこれがなりたつ。そこで、u から出発して、 p と同様の閉路 q を考える。そして、 v から出発し、 p の u までの部分を通り、u から q を通り、u へ戻って、再び p の残りの部分を通って v に至る道 p+q を得る。これが全体であれば終わり。そうでないなら、同じことを繰り返す。枝の個数は有限なので、必ず終わる。QED



閉路という条件を緩和したものも考えられる。すなわち、すべての枝をちょうど 1 回ずつ含む道が存在するかどうかを問題にする。一筆書きで始点と終点が異なってもよいとするのである。このときの必要十分条件はつぎの様になる。

系 2.2　　連結無向グラフ G=(V,E)において、すべての枝をちょうど 1 回ずつ含む道 p が存在するため

の必要十分条件は　次数が奇数の節点の個数が０または２個。

（証明）　必要性　ｐの始点と終点ｗ除けば、他の節点では定理の証明と同様に、次数が偶数であることがすぐわかる。

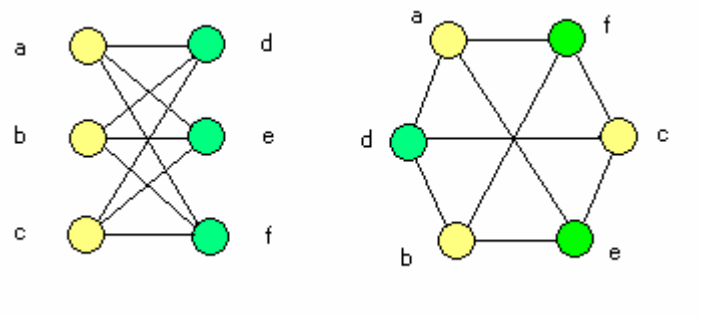
十分性　次数が奇数の節点の個数が０個なら、定理そのままである。２個とし、それを u,v とする。ｕとｖを端点とする新たな枝ｅを追加したグラフを考えると、これはすべての節点の次数が偶数であるから、オイラー閉路ｑが存在する。ｑよりｅを除いた道が求めるものである。

練習問題 2.1　ｎ≦5 を満たす各ｎについて、ｎ点からなるオイラー単純グラフの個数をすべて求めよ。

練習問題 2.2　有向グラフ G＝（V,E)は強連結とする。また、各節点について、入次数と出次数が等しいとする。このとき、各枝をちょうど１回ずつ通る有向閉路（有向オイラー閉路）が存在することを示せ。

３．グラフの同型と種々の例

まず、下の図をみてほしい。



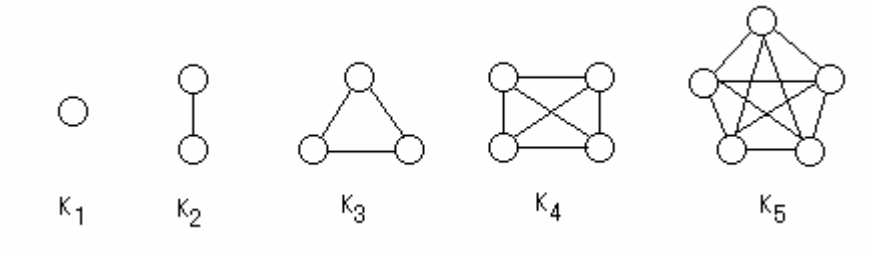
この２つの無向グラフは一見違うように見えるが、それは表現の仕方がちがうだけで、節点の個数が同じだけでなく、その結ばれ方も同じである。このようなグラフは同じものと思う。正確に述べると、無向グラフ G=(V,E) と G'=(V',E') に対して、V から V' への全単射写像 f と E から E'への全単射写像 g で、すべての e ∈E について、f(∂e)= ∂g(e)が成り立つような写像の組(f,g)を G=(V,E) と G'=(V',E') への同型写像という。ここで、∂は各枝に対して、端点を対応させる写像。すなわち、e の端点を u,v としたとき、∂e={u,v} である。G=(V,E) から G'=(V',E') への同型写像が存在するとき、G=(V,E) と G'=(V',E') は**同型**であるという。

有向グラフについても同様に同型写像が定義できる。違う点は∂e=(u,v)で、u が始点、v が終点と言う点だけである。

典型的な無向グラフの例をいくつか紹介する。

例１　完全グラフ K_n

これは節点の個数が n で、任意の異なる２節点に対して、それを端点とする枝がちょうど１つ存在する無向グラフを完全グラフ(complete graph)という。枝の個数は n(n-1)/2 となる。



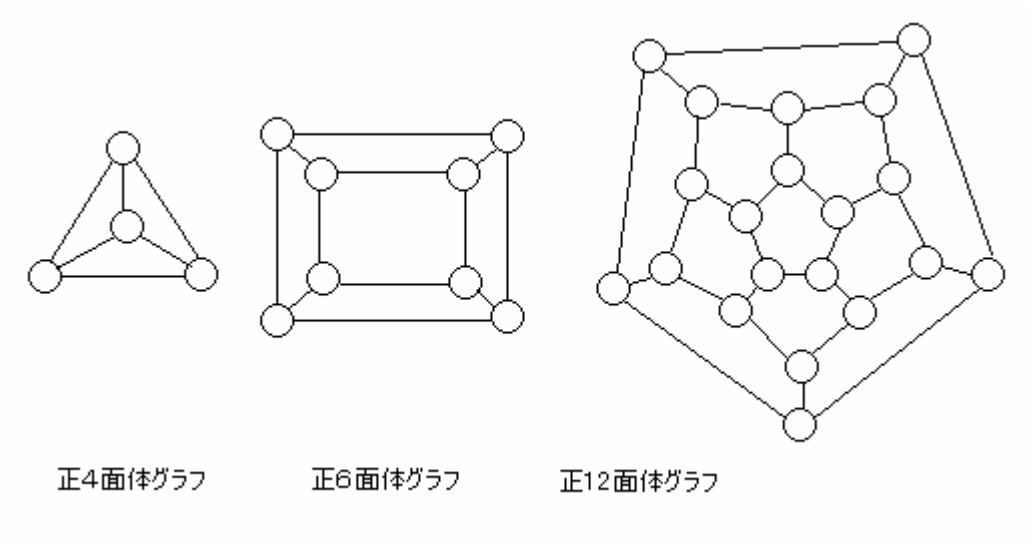
例２　完全２部グラフ　$K_{m,n}$

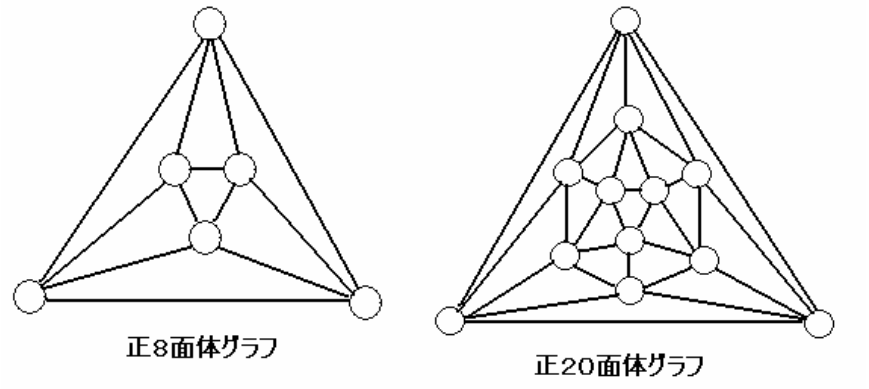
節点の集合 V が２つの部分集合 V_1 と V_2 に分かれていて、($V= V_1 \cup V_2$, $V_1 \cap V_2 = \phi$) 任意の枝について、その端点の一方は V_1 に他方は V_2 に属するとき、このグラフを２部グラフ(bipartite graph)という。$|V_1|=m$, $|V_2|=n$　で任意の V_1 の要素と任意の V_2 の要素に対して、それらを端点とする枝がち

ょうど 1 つあるような 2 部グラフを完全 2 部グラフと言い、$\mathbf{K_{m,n}}$ と書く。同型の定義のところであげた例は $K_{3,3}$ である。

例３　正多面体グラフ

正多面体は 5 通り存在して、それは正４面体、正６面体、正８面体、正 12 面体、正 20 面体である。これらの頂点を節点、辺を枝と思うと無向グラフができる。これらを**正多面体グラフ**という。平面の上に表現すると、次の図のように書ける。





さて、正 12 面体グラフと関わって、次のような問題がある。1859 年に数学者ハミルトンは正 12 面体の各頂点にブリュッセル、フランクフルトなど都市の名前をつけ、この正 12 面体の辺に沿って各都市をちょうど 1 回だけ訪れる旅路を求めよというパズルを発表した。グラフの言葉でいうと、各節点をちょうど 1 回含む閉路を求めよということになる。このような閉路をハミルトン閉路という。オイラー閉路は各枝をちょうど 1 回含む閉路であるが、節点は 1 回以上出て来てもよかった。今度は通らない枝があってもよいが、節点はすべてちょうど 1 回訪れなければいけない。上の正 12 面体グラフを見ながら、考えてほしい。

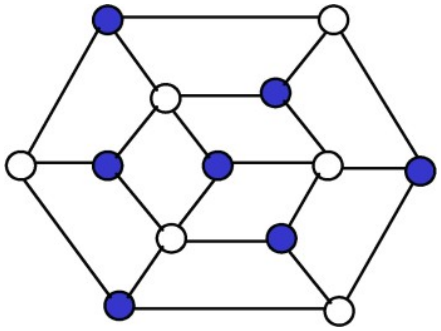
ハミルトン閉路を求める問題をもう１題。チェスのナイトの問題である。チェスの盤は８×８の６４の目から出来ているが、任意の位置から始めて、ナイトの動き方にしたがってすべての目にちょうど１回だけ留まるような動かし方を求めよ。これはチェスの目を節点、ナイトの１回の動き方で移動できる目同士が枝で結ばれていると考えれば、１つの無向グラフができる。そのグラフに対するハミルトン閉路を求める問題に他ならない。

一般的なグラフに対して、ハミルトン閉路が存在する条件を見つけるのは難しい問題である。しかし、次のことはすぐわかる。

定理 3.1　Ｇ＝（Ｖ，Ｅ）が 2 部グラフで、｜Ｖ｜が奇数であるとき、Ｇにハミルトン閉路は存在しない。

証明　Ｖ＝Ｕ∪Ｗ　Ｕ∩Ｗ＝Φ　と分けることができ、すべての枝はＵの節点とＷの節点を繋いでいる。従って、閉路は偶数個の枝からなり、同時に、偶数個の節点からなる。ハミルトン閉路が存在するとすれば、それは｜Ｖ｜の節点からなる閉路なので、これは矛盾である。

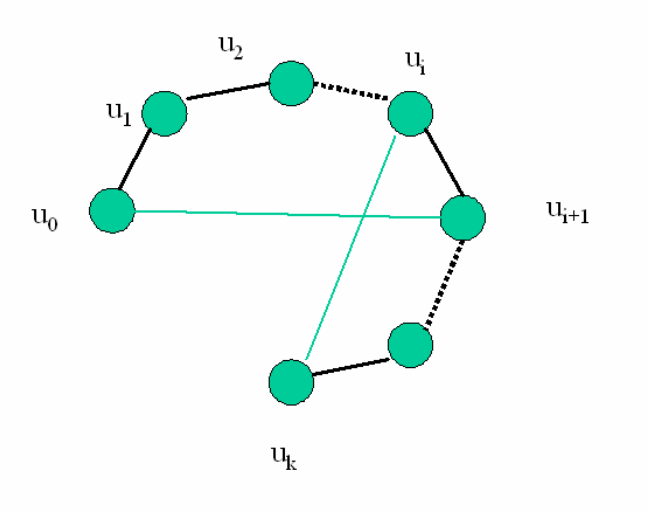
例



定理 3.2（ディラック 1952）

G=(V,E) は(単純）無向グラフで、ｎ＝｜Ｖ|≧３。最小次数≧ｎ/2　とする。Ｇ にはハミルトン閉路が存在する。

証明　まず、Gは連結である。というのは、もし、非連結とすると、節点が最小の連結成分の節点の個数はｎ/2 以下だから、単純性より、各点の次数はｎ/2－１以下となり仮定に反するからである。さて、Gにおける単純で最長な道をｐ：：$u_0u_1u_2\cdots u_k$　とする。$u_0u_1u_2\cdots u_k$はすべて異なるので、ｋ＋１≦ｎである。最長性よりu_0の隣接節点とu_k の隣接節点はすべてｐに含まれる。集合｛０≦ｉ≦ｋ－１｜u_{i+1}はu_0の隣接節点｝、｛０≦ｉ≦ｋ－１｜u_iはu_kの隣接節点｝はその要素がどちらもｎ/2 以上ある。ｋ＜ｎなので、必ず共通部分が存在する。すなわち、u_{i+1}はu_0の隣接節点で、しかも、u_iはu_kの隣接節点となる ｉがとれる。そこで、閉路ｃ：($u_0\,u_{i+1}$) ｐ[u_{i+1}、u_k] (u_k、、u_i) ｐ[u_i、u_0]を考える。ここで、ｐ[ｕ　、ｖ]は道ｐのｕからｖへの部分を表す。



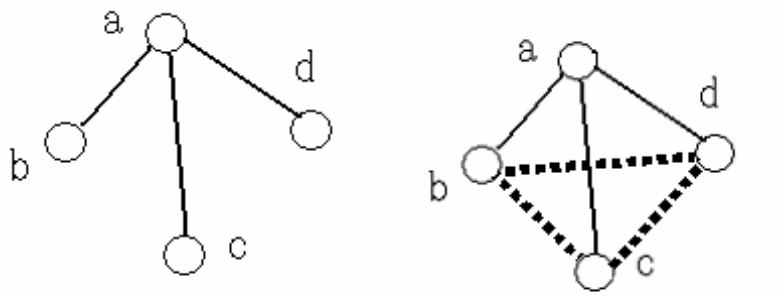
このｃが求めるハミルトン閉路である。もし、ｃに含まれない節点があれば、Gは連結なので、ｃにある節点u_tと隣接するｃに属さない節点ｖが存在する。ｖを始点とし、（ｖ、u_t）からｃをまわり、u_tの手前まで行く単純な道を考えると、それは長さがｋ＋１で、もとのｐより長くなるので矛盾。よって、ｃはハミルトン閉路である。証明終わり。

単色３角形問題

枝が２色に塗り分けられている無向グラフＧがある。Ｇに同じ色からなる３辺を持つ三角形（これを単色３角形）という。ここでは、次を示す。

定理 3.3　　n ≧６　のとき、K_n は単色 3 角形を含む。

証明　K_n はK_6 を部分グラフとして含むので、n=6 の場合に示せばよい。1 点a を固定して、a を端点とする枝は５本あるが、そのうち少なくとも 3 本は同色である。その枝をe_1，　e_2，　e_3とする。また、それぞれの端点を　b，c，d　とする。これらをつなぐ枝の１つがe_1，　e_2，　e_3と同じ色の場合はそこに単色３角形ができる。そうでない場合はb，c，dでできる３角形が単色３角形である。



例えば、パーティーに招待された n 人について、知り合いの場合は実線でつなぎ、そうでない場合は破線でつなぐと 2 色に塗り分けられた完全グラフができる。6 人以上の場合は
互いに知り合いである 3 人、もしくは、まったく知り合いでない 3 人のいずれかが存在することが言える。

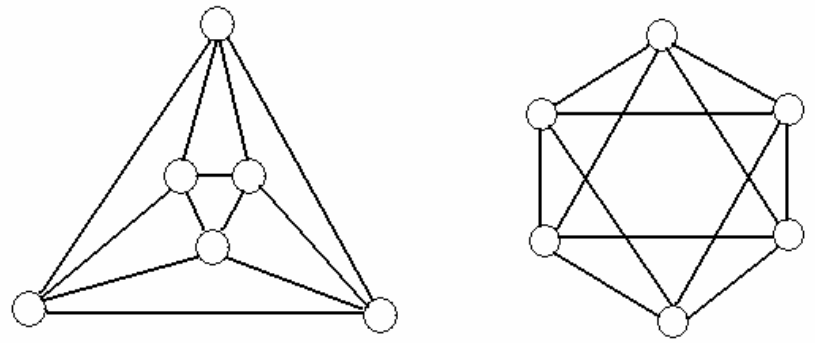
練習問題 3.1　正 12 面体グラフのハミルトン閉路を求めよ。

練習問題 3.2　本文で述べられたチェスのナイトの問題の解を求めよ。

練習問題 3.3　2 部グラフにおける閉路は偶数個の枝からなることを示せ。

練習問題 3.4　$G=K_4, K_5, K_{2,3}, K_{3,3}$ のそれぞれについて、$\kappa(G)$, $\lambda(G)$ を求めよ。

練習問題 3.5　次の２つのグラフは同型かどうか調べよ。



練習問題 3.6

(1)　3 点からなる有向（単純）グラフをすべて求めよ。

(2)　4 点からなる有向（単純）グラフをすべて求めよ。

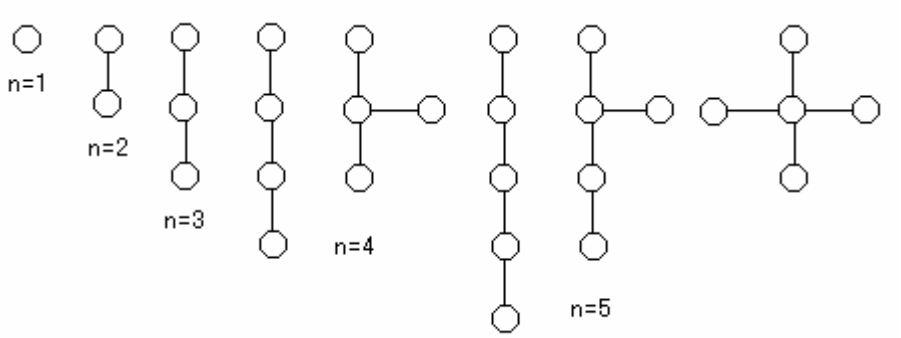
練習問題 3.7

オイラーグラフはハミルトン閉路を持つか？

４．木

閉路において、始点と終点を除いて、すべての節点が相異なるとき、単純閉路という。<u>以下、とくに断らないかぎり、閉路は単純閉路のことである。</u>

閉路を持たない連結無向グラフを**木(tree)**という。



定理 4.1

無向グラフ G＝(V,E)について次は同値である。

① G が木である。

② G に閉路がなく、|E|=|V|－1

③ G は連結で、|E|=|V|－1

④ G は連結で、枝を１つ除くと非連結になる。

⑤ G において、任意の２節点に対して、それらを結ぶ道が存在してしかも唯１つ。

⑥ G に閉路はないが、G の任意の２節点を結ぶ新しい枝を付け加えると閉路が１つできる。

証明

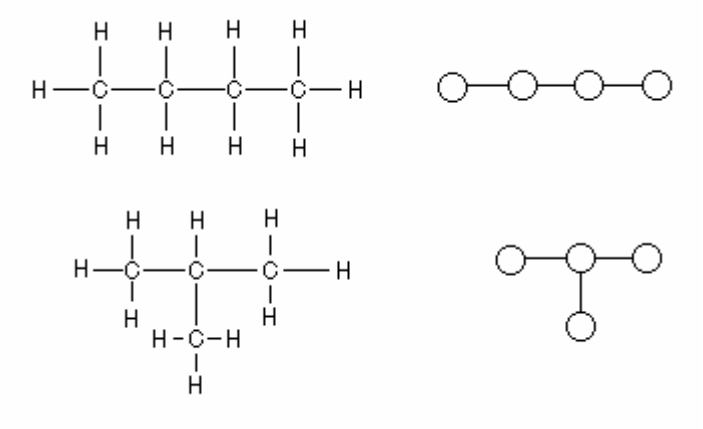
これらの同値性を順に見て行こう。ｎ＝|V|とおく。**①⇒②**はｎに関する数学的帰納法で示す。ｎ＝１のときは明らか。一般の場合は１つの枝ｅを取り除くと、２つの木に分解するが、それぞれに数学的帰納法の仮定を適用すればよい。**②⇒③**　G の連結性を示せばよい。もし、非連結なら各連結成分は木なので、すでに示した①⇒②を使って、枝の個数は節点の個数より１少ない。連結成分が 2 以上あると全体で、|E|＜|V|－1 となり矛盾が生じる。

③⇒④　枝を１つ除くと枝の個数はｎ－２となる。ｎ個の節点を連結にするには最低ｎ－１個の枝が必要だから、これは非連結。**④⇒⑤**　連結なので任意の２節点に対して、それらを結ぶ道が存在する。もし、２つ存在するなら閉路ができるので、枝を１つ除くと非連結になるという仮定に反する。**⑤⇒⑥**　閉路があれば、その閉路に含まれる２節点は２通りの道で結ばれることになり仮定に反する。よって、閉路はない。２節点を結ぶ新しい枝を付け加えると、この２節点はすでにある道で結ばれているので、それと新しい枝をあわせると閉路ができる。できる閉路が２つであるとすると、この新しい枝を取り除いても閉路があることになり、矛盾。⑥⇒①連結性を言えばよい。任意に２節点について、それを結ぶ枝

を付け加えると閉路ができるということは、その枝を付け加える前からその２節点をむすぶ道があるということなので連結性がいえた。QED

応用例

化学式 C_nH_{2n+2} で表される分子はいくつかの異性体を持つ。炭素原子の配置が決まると水素原子の位置は自動的に定まるので、炭素原子を節点と思い、それらの繋がり方をグラフで表すとｎ節点の木ができる。それで、ｎ節点の木が（同型をのぞいて）いくらあるかという問題に帰着される。この問題は 1875 年に Cayley によって解かれた。



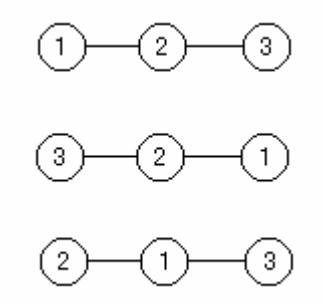
図はブタンとイソブタン。

練習問題 4.1　n≦7 のとき、ｎ点からなる木をすべて求めよ。

練習問題 4.2　Ｔ＝（Ｖ，Ｅ）は木とする。また、｜Ｅ｜≧１とする。このとき、Ｔには次数１の節点が２つ以上存在することを示せ。

５．ラベル付き木

ｎ個の節点を持つ木 T=(V,E)において、各節点に１からｎまでのラベルをつけて、区別したものをラベル付き木という。すなわち、α を集合{1,2,…,n}から V への全単射写像とし、木とこのような写像の組(T, α)をラベル付き木という。２つのラベル付き木(T, α)と(T', α')の同型写像は木の同型写像(f,g)で、しかも、ｆ α＝α'を満たすものをいう。

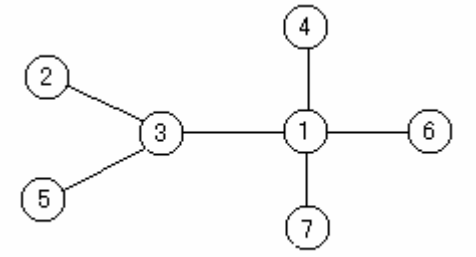


図で１番上のものと２番目は同じラベル付き木であるが、１番下のものはちがう。この図のように３節点からなるラベル付き木の場合は真中の節点に入れる数字で定まるので全部で３通り。一般にｎ節点のラベル付き木は何通りあるだろうか？つぎの定理がそれに答える。

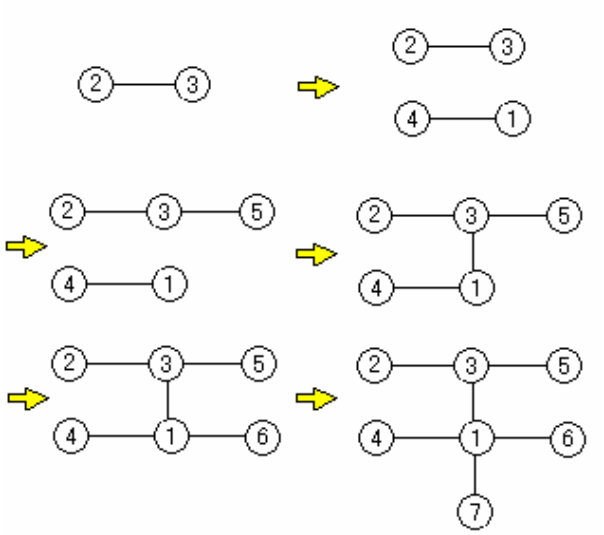
定理 5.1（Cayley 1889 年） ｎ節点の異なるラベル付き木は n^{n-2} 個ある。

証明　n=1,2 のときはラベル付き木は 1 通りなので定理は成り立つ。そこで、ｎ≧３とする。S={1,2,…,n }とし、S のｎ－2 個の直積を X とする。ラベル付き木 T に対して、つぎの様に X の要素$(a_1,a_2,\cdots,a_{n-2})$を対応させる。T の次数１の節点のうち、ラベルが最小のものを b_1 とし、b_1 に隣接している節点のラベルを a_1 とする。節点 b_1 とその隣接枝を取り除く。この操作を a_{n-2} が求まるまで繰り返す。逆に X の要素$(a_1,a_2,\cdots,a_{n-2})$に対して、ラベル付き木 T をつぎのように構成する。$a_1,a_2,\cdots,a_{n-2}$ のいずれとも異なる S の要素のうち最小のものを b_1 とし、a_1 と b_1 を枝でむすぶ。残りの数列 $a_2,\cdots,a_{n-2}$ および、S－{ b_1}について同じことを繰り返す。最後に、残った２つの数字を枝で結べばラベル付き木 T ができる。この対応が互いに逆写像になっていることはすぐにわかる。

例

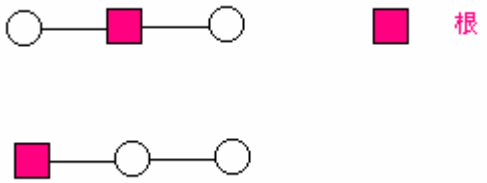


次数１の節点は２、５，４、７、６だから $b_1=2$、$a_1=3$、これを繰り返し、$b_2=4$、$a_2=1$、$b_3=5$、$a_3=3$、$b_4=3$、$a_4=1$、$b_5=6$、$a_5=1$ となる。よって対応する数列は
(3,1,3,1,1)となる。逆にこの数列からラベル付き木を構成していく過程を図示すると、

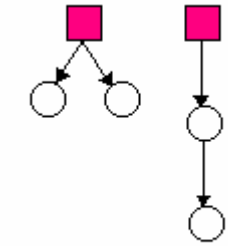


６．根付き木（rooted tree）

木において、１つの節点を特に指定したものを根付き木という。正確にかくと、木 T とその１つの節点ｒの組（T, r ）のことである。２つ根付き木 (T, r) (T', r')の間の同型写像は無向グラフとしての同型写像(f,g)で f(r)=r'をみたすもののことである。

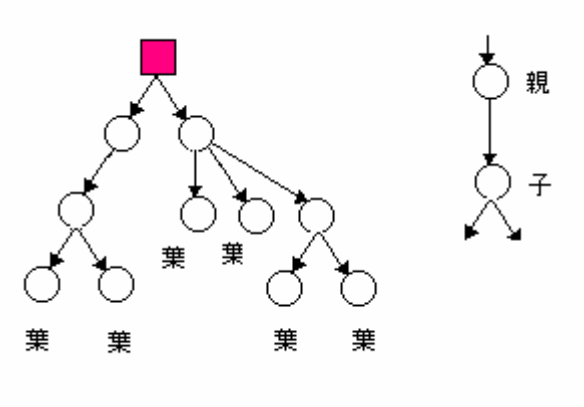


図は３節点からなる根付き木である。この２つは木としては同じものであるが、根付き木としてはちがうものである。（根の次数がちがう）木であるから、根より各節点に一意的に道が存在する。この道の向きにしたがって各枝に向きをつけると、根付き木は有向グラフとみることができる。上の図の根付き木に向きをつけ、根を一番上に書くと、



となる。（有向グラフと見て、）根付き木の隣接した節点について、枝の始点の方を親、終点の方を子という。子を持たない節点を葉、２つの節点 u,v について、u から v への有向路が存在するとき、u は v

の先祖、v は u の子孫という。
混乱がなければ、枝の矢印は省略し、また根も○で書く。

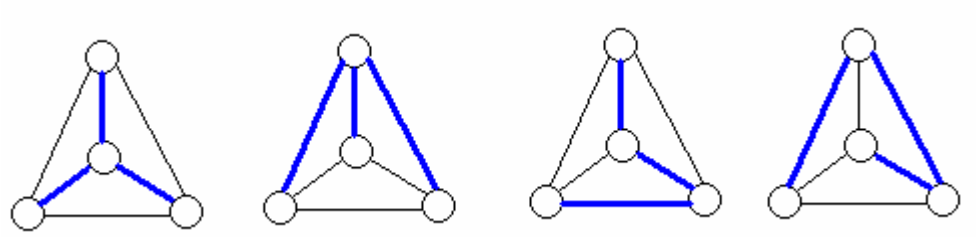


練習問題 6.1　n=3, 4, 5, 6 について、ｎ点からなる根付き木をすべて求めよ。

７．グラフの生成木

連結無向グラフ G=(V,E)の部分グラフ T=(V’,E’)が V=V’でしかも T 自身は木のとき、T をＧの**生成木**(spanning tree)または単にＧの木という。グラフＧ=(V,E)の生成木は V= V’を満たす連結部分グラフ T=(V’,E’)の中で極小なものと言える。また、閉路をもたない部分グラフの中で極大なものとも言える。

例



１つのグラフにどれだけ生成木があるかという問題に関して、次のような定理がある。

定理 7.1.　完全グラフK_nの生成木の総数はn^{n-2}である。

（証明）　K_nの節点に１からｎまでのラベルをつける。K_nの生成木はラベル付き木であるし、逆に完全グラフであるから、任意に２節点は枝で結ばれている。したがって、任意のラベル付き木はK_nの生成木である。よって、Cayley の定理より。

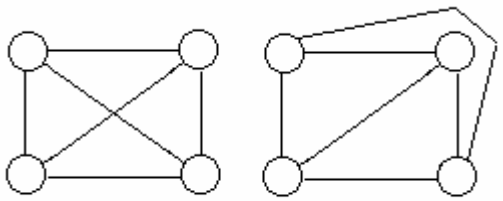
さて、一般の連結無向グラフ G=(V,E)にもどって、G の生成木 T=(V,E’)に対して、E－E’を T の補木という。e∈E－E’を T に付け加えると、第 4 節で示したように、閉路が１つできる。この閉路を構成する枝の中から、e 以外の枝を取り除く。閉路から１つ枝を取っても連結性はたもたれるし、閉路がなくなったので、新しくできた部分グラフはＧの生成木である。このように、１つの生成木からべつの生成木を作る操作を木の初等変形という。

定理 7.2.　　連結無向グラフ G=(V,E)の２つの生成木 $T_1=(V,E_1)$と $T_2=(V,E_2)$は何回かの初等変形で移りあう。

（証明） $E_2=E_1$ なら何もすることはない。そこで、E_1 に含まれないの枝 E_2 の枝 e をとってきて T_1 を初等変形する。つまり、T_1 に枝 e を付け加えると閉路ができる。（E_2 は閉路をもたないので）この閉路を構成する枝すべてが E_2 に属することはない。そこで、**E_2 に属さない枝をこの閉路から除去し**、新しい木 T’を作る。すると、T’と T_2 は変形前より共通枝が１つ増えている。したがって、この操作を繰り返せば、かならず、T_2 に達する。

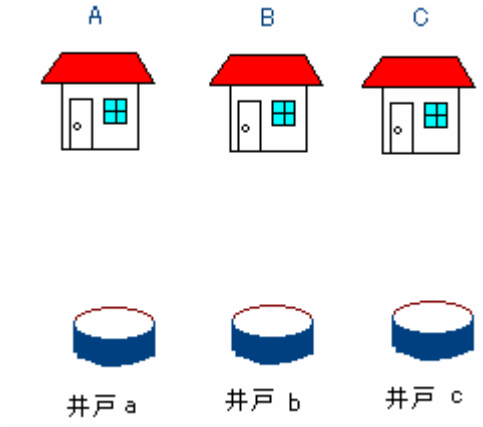
８．平面グラフ

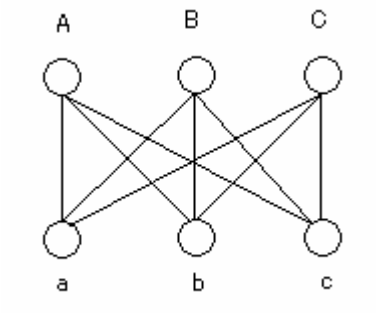
無向グラフが与えられたとき、それが枝が交差することなく平面上で表現できるかという問題を考える。簡単な例から始める。



図の左のグラフは枝が交差しているが、右のように書き換えると交差しない。勿論、この２つは同じグラフである。このように枝が交差しないように平面上に実現されたグラフを平面グラフという。平面グラフと同型なグラフを平面的グラフという。さて、次の有名なパズルを考えてみる。

供給問題　３軒の家と３つの井戸がある。井戸がしばしば乾いてしまうので、どの家からもそれぞれ３つの井戸に行かなければならない。３軒の家の住人はそれぞれ仲が悪く、顔も会わせたくない。そこで、各家から各井戸へ途中で絶対交わらない道を作ることにした。どのように道を作ればよいか？立体交差はだめ。





家を A,B,C 井戸を a,b,c とすると右のようなグラフができる。これはすでに登場した $K_{3,3}$ である。この枝は現在交差しているが、うまくやれば交差しないで書けるかということになる。別の言い方をすれば、$K_{3,3}$ は平面的グラフか？という問題になる。

そこで、グラフが平面的グラフであるための必要条件を求めてみる。平面グラフ G=(V,E)により、平面はいくつかの領域に分割されるが、この領域を面と呼ぶことにし、その個数を f とする。また、n=|V|, m=|E|とおく。前の例の正４面体グラフなら、f=4, n=4, m=6 で、f－m+n=2 となる。また、正 12 面体グラフなら、f=12, n=20, m=30 で、f－m+n=2 である。

定理 8.1（オイラーの公式）　連結平面グラフ G に対して、f－m+n=2 が成り立つ。
（証明）枝の個数に関する数学的帰納法で示す。ｍ＝０のとき、連結なので、n＝１で f ＝１。よって、このとき成立。次にｍ－１以下の枝をもつ連結平面グラフについてこの公式が成り立つと仮定して、G がｍ個の枝を持つ場合を証明する。G に閉路がない場合は G は木なので、m= n-1 また、 f=1 だから、f －m+n=1－(n－1)+n=2 となり、成立。G に閉路がある場合、閉路に含まれる枝の１つを e とする。G から e を取り除いたグラフ G－e は連結な平面グラフで、面の個数ｆ－１、枝の個数はｍ－１、節点の個数はｎである。数学的帰納法の仮定より、G－e については公式がなりたつので、（ｆ－１）－(ｍ－１)+n=2 .したがって、f－m+n=2 が言えた。

系 8.2　(1)連結単純平面的グラフ G がｎ個（ｎ≧３）の節点とｍ個の枝を持つとき、
ｍ≦３ｎ－６が成り立つ。
(2) さらに、G に 3 角形がないなら、ｍ≦２ｎ－４が成り立つ。

（証明）示したいことは節点の個数と枝の個数に関することだから、G は平面グラフとしてよい。このとき、できる面の個数を f とおく。単純だから、１つの面は最低 3 個の枝で囲まれている。よって、３ｆ≦2m. 一方、定理より、ｆ＝ｍ－ｎ＋２だから、3(ｍ－ｎ＋２)≦2m.　これを整理して、（１）が言えた。（２）については条件より、各面は最低４つの枝で囲まれているので、４ｆ≦2m.あとは（１）と同様。

系 8.3　K_5　および　$K_{3,3}$　は平面的でない。

（証明）K_5 の場合ｍ＝１０、ｎ＝５である。もし平面的なら、系１（１）に矛盾する。$K_{3,3}$ の場合ｍ＝９、ｎ＝６である。また、$K_{3,3}$ では１つの閉路は最低 4 個の枝でできている。もし、平面的なら、系１（２）に矛盾する。

という訳で供給問題はいくら考えてもうまい道はつくれない。3 軒の家が仲良くすることが１番。

前述のように、正多面体は正４面体、正６面体、正８面体、正１２面体、正２０面体の５種類しか存在

しない。このことはピタゴラス（紀元前500年ころの数学者）も知っていたようである。ここでは定理（オイラーの公式）を用いて、証明しよう。

定理 8.4　正多面体は正４面体、正６面体、正８面体、正１２面体、正２０面体の５種類しか存在しない。

証明　対応する正多面体グラフで考える。節点の個数を ｎ、枝の個数を ｍ、面の個数を ｆとする。また、各節点の次数（これは正多面体なので一定）は ｘ（ｘ≧３）とし、１つの面はｙ個（ｙ≧３）の辺を持つとする。１つの面はｙ個の節点を持つので、全体として、ｆｙ個、しかし、１つの節点はｘ個の面に共通しているので、ｆｙ＝ｎｘ　が成り立つ。また、１つの面はｙ個の枝をもつので、全体としてｆｙ個、しかし、１つの枝は２つの面に共通しているので、ｆｙ＝２ｍ が成り立つ。この２つの式とオイラーの公式 ｆ－ｍ＋ｎ＝２から次のようにして、この定理が証明できる。２ｆ－２ｍ＋２ｎ＝４にｆｙ＝２ｍを代入すると、

２ｆ－ｆｙ＋２ｎ＝４　‥①

また、ｆｙ＝ｎｘだから、①から

２ｆ－ｎｘ＋２ｎ＝４　‥②

①＋②から　（４－ｙ）ｆ＋（４－ｘ）ｎ＝８　‥③

４－ｙ≦０かつ４－ｘ≦０なら左辺は≦０となるが、右辺＞０なので矛盾。よって、４－ｙ＞０　または４－ｘ＞０。つまり、ｙ≦３またはｘ≦３である。もともと、ｘ、ｙは３以上なので、ｘ＝３またはｙ＝３が成り立つ。一方、①×２＋②から、

２（３－ｙ）ｆ＋（６－ｘ）ｎ＝１２　‥④

３－ｙ≦０なので、６－ｘ＞０　つまり、ｘ≦５が成り立つ。①＋②×２より、

（６－ｙ）ｆ＋２（３－ｘ）ｎ＝１２　‥⑤

上と同様に、ｙ≦５が言えた。以上より、（ｘ、ｙ）は（３，３）、（３，４）、（３，５）、（４，３）、（５，３）の５通り。（ｘ、ｙ）＝（３，３）の場合、④に代入すると、ｎ＝４、⑤に代入すると、ｆ＝４がわかり、ｆｙ＝２ｍより、ｍ＝６。正４面体。（ｘ、ｙ）＝（３，４）の場合、⑤より、ｆ＝６．②に代入して、ｎ＝８．ｆｙ＝２ｍより、ｍ＝１２。正６面体。

（ｘ、ｙ）＝（３，５）の場合、⑤より、ｆ＝１２とわかり、ｎ＝２０、ｍ＝３０となる。正１２面体。

（ｘ、ｙ）＝（４，３）の場合、④より、ｎ＝６。①に代入して、ｆ＝８。

ｍ＝１２となる。正８面体。最後に、（ｘ、ｙ）＝（５，３）の場合、④より、ｎ＝１２．

ｆ＝２０．ｍ＝３０．正２０面体。以上で、定理が証明できた。

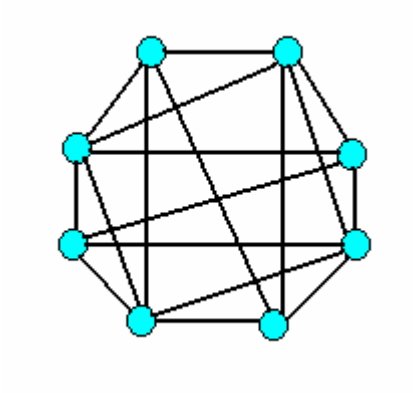
証明より、正多面体の頂点の個数などがわかったので、まとめておく。

	面の個数 f	辺の個数m	頂点の個数 n	x	y
正４面体	４	６	４	３	３
正６面体	６	１２	８	３	４
正８面体	８	１２	６	４	３
正１２面体	１２	３０	２０	３	５
正２０面体	２０	３０	１２	５	３

この表をながめていると、正６面体と正８面体は面の個数と頂点の個数が入れ替わっていることに気がつく。同じことは、正 12 面体と正２０面体にも言える。このことは幾何学的に次のように説明できる。正６面体（立方体）の各面（正方形）の中心に赤い点を書く。赤い点は６つできる。これらを直線（線分）でつなぐ。すると、正８面体ができる。正８面体から同じことをやると、正６面体ができる。双対性（duality)ですな。

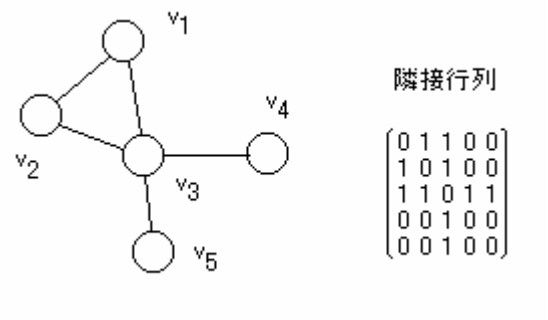
練習問題 8.1　K_n　(n≧5)は平面的ではないことを証明せよ。

練習問題 8.2　次のグラフは平面的かどうか調べよ。

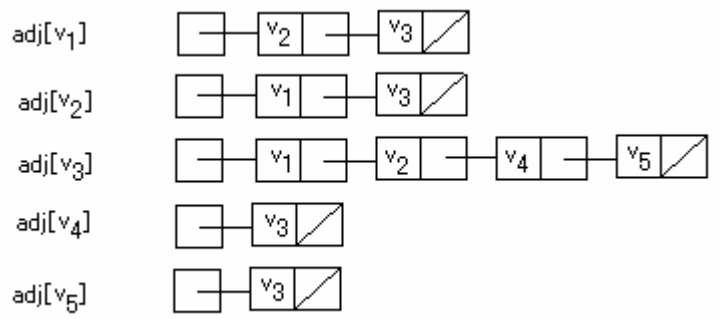


９．隣接行列と隣接リスト

グラフG=(V, E)に関わる諸問題を計算機を用いて解決するためにはその解決のためのアルゴリズムとともに如何に効率よく計算機にG=(V, E)を格納するかという点が肝要となる。
グラフは単純と仮定する。計算機に 2 次元配列でグラフを記憶させる方法として、次の**隣接行列**を用いる。まず、無向グラフG=(V, E)の場合であるが、n=|V|とし、V＝{v_1,v_2,…,v_n}とする。v_i と v_j を結ぶ枝があるとき、a_{ij}=1,そうでないとき、a_{ij}=0 と決める。このとき、n 次正方行列　M＝（a_{ij}）をG=(V, E)の隣接行列という。有向グラフG=(V, E)の場合は v_i を始点 v_j を終点とする枝があるとき、a_{ij}=1, そうでないとき、a_{ij}=0 と決める点が異なるだけである。
次に隣接リストであるが、これは連結リストというデータ構造を用いる。



adj[u]={ u に隣接している節点 }
とおく。この集合を順番は適当にきめて、リストでつなぐ。上のグラフの隣接リストは

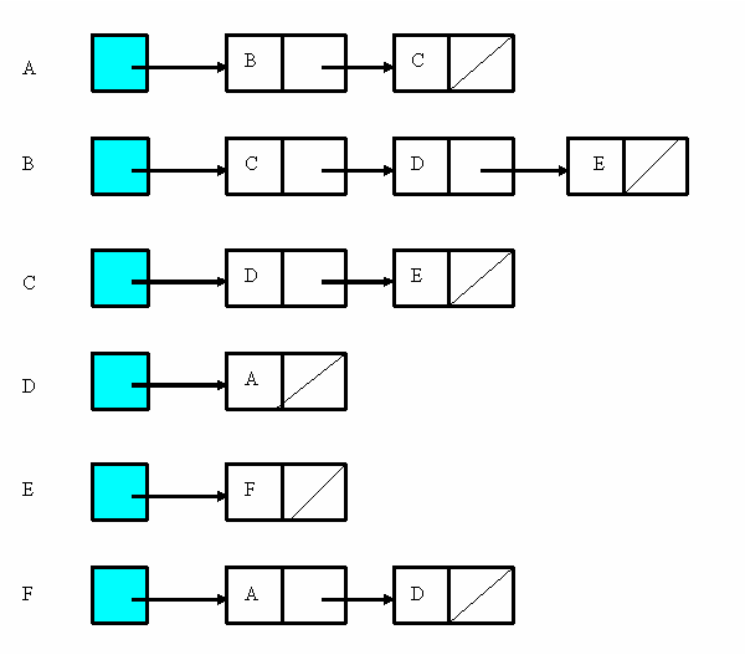


となる。隣接行列では n ^{2} 個の記憶場所が必要なのに比較して、隣接リストでは記憶場所は 2|E| となっている。

練習問題 9.1

(1)　正 4 面体グラフの隣接リストを書け。
(2)　完全 2 部グラフ　$K_{3,2}$ の隣接リストを書け。
(3)　完全グラフ　K_5 の隣接行列を書け。

練習問題 9.2　次の隣接リストで表される有向グラフ G について、設問に答えよ。

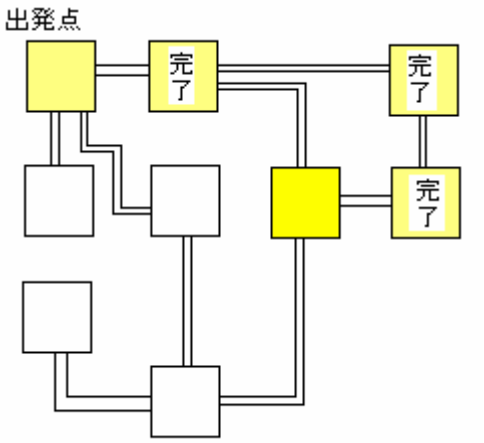


(1)　このグラフ G を書け。
(2)　G の（単純な）有向閉路をすべて求めよ。
(3)　G は強連結か？
(4)　G の隣接行列を求めよ。
(5)　Gの枝の向きをすべて反対にしたグラフG^TをGの転置グラフという。G^Tの隣接リストを求めよ。
(6)　Gの隣接行列とG^Tの隣接行列はどのような関係にあるか？

１０．グラフにおける探索

グラフおける探索はある節点を出発点として、そこから枝にそって次の節点へという様に進んでいく。有向グラフでは枝の向きにしたがって進む。節点を部屋、枝を部屋と部屋をつなぐ通路と思うと、探索の目的は部屋の明かりをつけることとしょう。出発点の部屋から明かりをつけ、つぎに行ける部屋はどこかと考える。隣接リストを見るわけだ。そして、次の部屋へと進む。行き止まりになると戻ることも考える必要がある。出発点から行ける部屋すべてに明かりをつければおわりであるが、よーく整理しないと忘れた部屋ができてしまう。それでは困るので、明かりのついている部屋を 2 通りに分類する。そ

こに隣接している部屋のすべてにすでに明かりがついている部屋と隣接している部屋の中にまだ明かりのついてないものがある部屋の２種類に。前者には完了という札でも貼っておくとよい。そうすると、完了の札がある部屋では隣接している部屋にもう行く必要がないことがすぐわかる。さて、例え話はこのくらいにして、ある節点に探索がおよんだ時、その節点は既探索節点という。その節点に隣接している節点すべてを探索し終えたとき、その節点を走査済み節点という。グラフにおける探索の途中において、節点は未探索節点、既探索走査未完節点、既探索走査済み節点の 3 種類に分かれる。



さて、節点をどのような順序で探索を進めるかであるが、代表的な方法が２つある。それは**広さ優先探索**と**深さ優先探索**と呼ばれているもので、広さ優先探索は出発点からそれに隣接する節点を全部探索し、つぎにその中で最初にしらべた節点の隣接節点を全部探索し、…というように言わば几帳面な方法。深さ優先探索は出発点からそれに隣接する節点に進み、その節点に隣接する節点があれば、そこに進むというような、「進めるだけ進め」タイプの方法である。勿論、進めなくなれば、ひとつ戻るわけであるが。それでは正確に探索のアルゴリズムを書くことにする。グラフG=(V,E) ,s は出発点となる節点。Q はキュー。

v∈Ｖについて、color[v]に未探索のときは white, 既探索走査未完の時は gray,
既探索走査済みの時は black を入れることにする。また、π[v]にはｖがどの節点から最初に走査されたか記録しておくことにする。なお、グラフは有向でも無向でもよい。

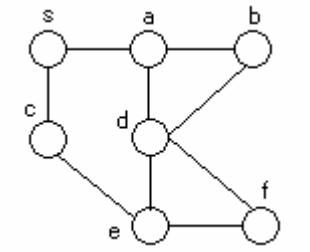
広さ優先探索(Breadth First Search)

BFS(G,s)

```
for each   v∈V  do
           color[v]←white
           d[v]  ←∞
           π[v]  ←  nil
color[s]  ←gray
d[s]←0
```

```
Q←{ s }
while   Q≠φ  do
    v←Head(Q)
    for each u∈adj[v] do
        if color[u]=white   then
            color[u] ←gray
            d[u] ←d[v]+1
            π[u] ← v
            u をQに入れる
     Dequeue(Q,v)
     color[v] ←black
```

例。つぎの無向グラフで BFS を実行してみる。



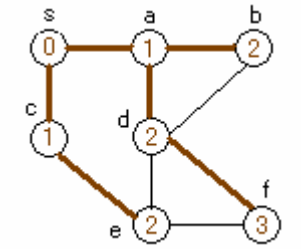
まず、出発点 s が Q に入り、Dequeue(Q)で s が出てくる。adj[s]={a, c}だから、これらが Q に入る。つぎに Q から出てくるのは a で adj[a]={s, b, d}だがこのうち color が white の b, d が Q に入る。このとき、Q＝[c, b, d]となっている。だから、次に出てくるのは c で、Q に e が入る。つぎに Q から b が出てくるが、白い隣接節点はないので、次へ進み、d が Q から出てくる。それで、f が Q に入り、あとは e, f の順に出てきて、Q は空っぽになって終わり。実行終了後の各変数の値はつぎのとおり。

	s	a	b	c	d	e	f
d[ ]	0	1	2	1	2	2	3
π[ ]	nil	s	a	s	a	c	d

部分グラフ $G_\pi=(V_\pi, E_\pi)$を

$V_\pi=\{ v \in V \mid \pi[v]\neq nil\}\cup\{s\}$, $E_\pi=\{(\pi[v],v)\mid \pi[v] \neq nil \}$

によって定めと、これは木になる。**BFS 木**という。上の例の場合はつぎのようになる。なお、



節点の中の数値は d[v]の値で、これは BFS 木における s から v へ至る道の枝の個数に一致する。

深さ優先探索(depth first search)

色が gray になった時刻を記憶する d[ ]と色が black になった時刻を記憶する f[ ]を用意する。再帰呼び出しを用いて、深さ優先探索は次のように記述できる。

DepthFirstSearch

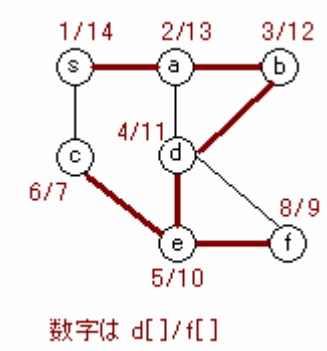
```
for each v ∈V  do
    color[v]←white
    π[v]←nil
time ←0
for each v ∈V  do
     if color[v]=white then
            DFS(G,v)
```

DFS(G,v)

```
color[v] ←gray
d[v] ←time←time+1
for each u ∈adj[v]
     if color[u]=white then
          π[u]←v
          DFS(G,u)
f[v] ←time←time+1
color[v] ←black
RETURN
```

広さ優先探索の場合と同様に、部分グラフ $G_\pi=(V_\pi, E_\pi)$ を
$V_\pi=V$, $E_\pi=\{(\pi[v],v)\mid \pi[v]\neq nil\}$
によって定めと、これは森になる。**DFS 森**という。いくつかの木の集まりになっている。

例　広さ優先探索のときと同じグラフで深さ優先探索を実行してみる。前の例と比較するために、for each v ∈V の順番は s からとする。すると、最初に color[s]←gray, d[s] ←1, そして、 DFS(G, s)が呼ばれ、for each u ∈adj[s]になる。この順番はアルファベット順とすると、最初は a で、color[a]←gray、d[a] ←2、π[a] ← s となり、DFS (G, a) が呼ばれる。それで、for each u ∈adj[a]となる。このあと探索される順だけ書くと、b, d, e, c
となる。c の隣接節点に white はないので、f[c] ←7, color[c] ←black となって、ひとつ前に戻り、つぎに f を探索することになる。そのあと、 s まで戻って終わり。



定理 10.1　G＝（V，E）に対して、深さ優先探索を実行したとする。その結果、任意の２節点 u, v に対して、次の３つの条件のいずれか１つが成り立つ。
（１）区間[d[u], f[u]]と区間[d[v], f[v]]は共通部分が空集合。u, v はＤＦＳ森において、一方が他方の子孫でも、先祖でもない。
（２）d[v] ＜ d[u] ＜ f[u] ＜ f[v]。u はＤＦＳ森において v の子孫である。
（３）d[u] ＜ d[v] ＜ f[v] ＜ f[u]。v はＤＦＳ森において u の子孫である。
証明：d[u]<d[v] かどうかで２通りに分け、さらに、それぞれを、f[u]　あるいは　f[v]の大小関係で２通りに分けると全部で４通りの場合が考えられる。
①　d[u]<d[v]<f[u]の場合。v が発見されたとき、u はまだ gray である。従って、v は u の子孫であることがわかる。アルゴリズムより、v が終わらないと u は終わらないので、f[v]<f[u]。
d[v] <　f[v] なので、（３）が成り立つ。
② d[u] <f[u] <d[v]の場合。d[v] <　f[v] なので、(1)が成り立つ。v が発見されたとき、u は終わっているので、子孫でも先祖でもない。（１）が成り立つ。
③ d[v]<d[u]<f[v] の場合、①と同様に、（２）が成り立つ。
④ d[v] <f[v] <d[u]の場合。②と同様に（１）が成り立つ。
証明おわり。

この定理より、次がわかる。

系 10.2　G＝（V，E）に対して、深さ優先探索を実行したとする。

ＤＦＳ森において、v が u の真の子孫である　⇔　d[u] ＜ d[v] ＜ f[v] ＜ f[u]

定理 10.3（White path theorem）

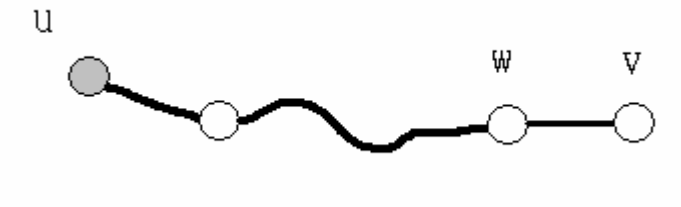
G＝（V，E）に深さ優先探索を実行するとする。

ＤＦＳ森において、v が u の子孫 ⇔u の発見時刻　d[u] において、u から v へ白い道が存在

ここで白い道とは途中の節点の色がすべて白（white）であるような道のことである。

証明：⇒：v が u の子孫だとする。ＤＦＳ森における u から v への道を考える。この道の各節点は u の子孫なので、系 10.2 より、u の発見時刻において、すべて白色である。

⇐ ：u から v へ白い道 p が存在するとする。背理法で示す。 v が u の子孫でないとする。必要なら v を取り替えて、p における v より u に近い節点はすべて u の子孫　であると仮定しても良い。w を p における v の１つ手前の節点とする。(u=w の可能性もある。）仮定により w は u の子孫。系 10.2 より f[w]≦f[u]。



v は w の隣接節点なので、v が終わらないと終わらない。すなわち、f[v] ＜ f[w]。よって、d[v]<f[w]である。時刻 d[u]には v は白だったので、d[u] < d[v] < f[w] ≦ f[u]。とくに、d[u] < d[v] < f[u]。定理 10.1 より、d[u] < d[v] <f[v] < f[u]であり、v は u の子孫。矛盾。

練習問題 10.1

練習問題 9.2 のグラフに対して、広さ優先探索を実行し、BFS 木を求めよ。但し、始点は A とし、for 文はアルファベット順に実行されるものとする。

練習問題 10.2

練習問題 9.2 のグラフに対して、深さ優先探索を実行し、DFS 森を求めよ。但し、for 文はアルファベット順に実行されるものとする。また、各節点 v について、d[v],f[v]を求めよ。

１１．深さ優先探索とトポロジー順序

有向閉路をもたない有向グラフは**アサイクリック・グラフ**という。英語の directed acyclic graph の頭文字をとって、dag と言われることもある。アサイクリック・グラフの節点はつぎの条件（＊）をみたすように順序付け（正確には全順序付け）することができる。これをトポロジー順序という。

（＊）(u,v) ∈E　⇒　u ＜ v

つまり、節点を 1 列に並べたとき、すべての枝は順序の小さい方の節点から大きい方の節点に繋がれているということである。

深さ優先探索をアサイクリック・グラフに適用すると、トポロジー順序が得られる。このことを説明しよう。

定理 11.1

G=(V,E)をアサイクリック・グラフとする。これに、DepthFirstSearch を実行すると、各節点 v に f[v]という走査完了時刻が記録される。このとき、

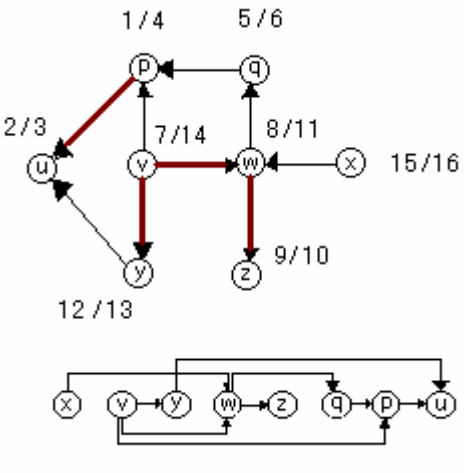
(u,v) ∈Ｅ　⇒　f[v] ＜ f[u]

が成り立つ。

証明：(u,v)という枝があれば、adj[u]に v が含まれているので、u が走査点のときに、v の色が white であれば、f[v]<f[u]であるし、v の色が gray であれば、これは v から u への有向路があることになり、これは G に有向閉路があることになり仮定に反する。また、u が走査点のときに、v の色が black であれば、v は走査が完了しているので、f[v]<f[u]。

この定理により、走査完了時刻 f[ ]の降順に節点を並べればトポロジー順序が得られる。

例１． つぎのアサイクリック・グラフに DepthFirstSearch を実行する。すると、図にすでに書き込んであるように、d[ ]/f[ ]が定まる。f[ ]の降順に並べると下図のようになる。



トポロジー順序を深さ優先探索を使わない方法で求めることもできる。それは、次のアルゴリズムによる。

TopSort(G)

if G≠φ　then

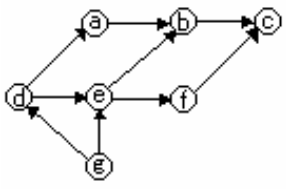
　　in-deg(v)=0 となる v を1つ見つけ、キューQ にいれる。

　　G から v を除去したグラフを G'とする。

　　TopSort(G')

練習問題 11.1　アサイクリック・グラフには必ず、入次数が 0 の節点が存在することを示せ。

練習問題 11.2　次の有向グラフのトポロジ順序を１つ定めよ。



１２．ネットワーク

G=(V,E)を有向または無向グラフとする。E 上の実数値関数 w と G の組を重み付きグラフあるいはネットワークという。wは重み(weight)という。冒頭にあげた情報通信網の他に、例えば節点を駅とし、枝をそれらを結ぶ路線、w としては所要時間や営業キロ数をとることも考えられるし、都市間を結ぶ道路とその輸送量を考えれこともできる。電気回路網で各導線を流れる電流を重さと考えることも可能である。本節において、ネットワークに関する重要なアルゴリズムについて順次紹介する。

12.1 最小木問題

重みｗがついた連結無向グラフ G=(V,E)の生成木 T=(V,F)に対して、T の重み w(T)を

$w(T)=\sum_{e\in F}w(e)$

と定義する。G の生成木の中で、重みが最小のものを最小木という。最小木は１つとは限らないが、その１つを求める問題が最小木問題である。この問題については２つの有名なアルゴリズムがあり、１つは Kruskal のアルゴリズム、もう１つが Prim のアルゴリズムである。それを説明する前に、両方の根源となっている一般的なアルゴリズムを述べる。

定義：A はある最小木の一部分をなす枝の集合とする。枝 e が A に対して、安全(safe)とは
A∪{e} もある最小木の一部分となっていることをいう。

```
Generic-MST(G)
A←φ
while　A は最小木ではない do
　A に対して安全な枝 e を見つける
 A←A∪{e}
return
```

このアルゴリズムで重要なことは A がまだ最小木になっていないときに、A はある最小木に真の部分集合なので、A に対する安全な枝は必ず存在する。問題はその安全な枝をどのように見つけるかという点にある。

無向連結グラフ G=（V,E)に対して、V の分割（S,V－S)をカットという。枝 e の端点の一方が S に属し、他方がが V－S に属すとき、枝 e はカット（S,V－S)を横断するという。枝の集合 A のどの枝もカット（S,V－S)を横断しないとき、カット（S,V－S)は A を尊重するという。

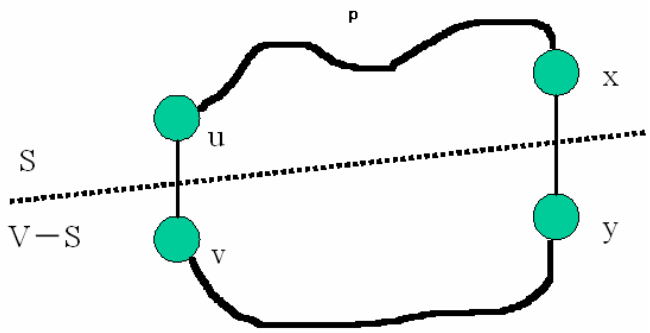
定理 12.1　G=（V,E)は重み w を持った無向連結グラフとする。A は G のある最小木に含まれる枝の集

合、(S,V－S)は A を尊重するカット、(u,v)は (S,V－S)を横断する枝の中で重みが最小のものとする。このとき、(u,v)は A に対して安全である。

証明　仮定より、A はある最小木 T に含まれる。(u, v)が T に含まれれば証明は終わり。そこで、(u, v) は T に含まれないとする。このとき、別の最小木 T´を見つけて、A∪{(u,v)} が T´に含まれることを言えば良い。T における u から v への道を p とする。(u, v)はカット (S,V－S)を横断する枝なので、節点 u と v はそれぞれ、このカットの反対側にある。従って、道 p はどこかでこのカットを横断しなければならない。その枝を(x,y)とする。そこで、木の初等変形をおこなう。つまり、T から(x,y)を除去し、(u,v)を付け加えて、新しい生成木 T´をつくる。

T´＝T ∪ {(u, v)}-{(x, y)}

T´が最小木であることを言おう。(u, v)は横断する枝の中で重みが最小であるから、
w(u, v)≦ w(x, y)、w(T´) ＝w(T)-w(x, y)+w(u, v)なので、w(T´) ≦w(T)。しかし、T は最小木なので、w(T´) ＝w(T)が成り立つ。つまり、T´が最小木であることが言えた。(S,V－S)は A を尊重するカットなので、(x,y)は A には属さない。だから、A は T´に含まれる。つまり、A∪{(u,v)} が T´に含まれることが言えた。証明終わり。



系 12.2　G= (V,E)は重みwを持った無向連結グラフとする。AはGのある最小木に含まれる枝の集合とする。Cは森G_A=(V,A)の 1 つの連結成分とする。(u,v)がCと他の連結成分を結ぶ枝の中で重みが最小であるなら、(u,v)はAに対して、安全である。

証明　C に属する節点の集合を S としてカットを考えれば、このカットは A を尊重するので、定理が適用できる。

さて、この系 12.2 を用いて、具体的にカットを指定し、それを横断する最小重みの枝をみつける 2 つのアルゴリズムを紹介する。まず最初はクラスカルのアルゴリズム。

MST_Kruskal

キューQ に G の枝を重みの小さい順に入れる。

A←ϕ

while |A|<|V|−1 do

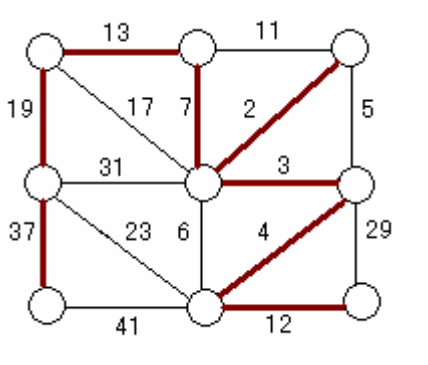
　　e←DEQUEUE(Q)

　　if　(A∪{e}が閉路を持たない) then

　　　　A←A∪{e}

このアルゴリズムで選ばれる枝は森G_A=(V,A)の異なる２つの連結成分を結ぶ枝の中で重みが最小のものである。閉路ができない⇔異なる連結成分を繋いでいる

例１．つぎの重み付きグラフについて、Kruskal のアルゴリズムを実行してみる。枝の横の数字はその枝の重みである。



次に Prim のアルゴリズムは１つの節点 s を選び、

MST_Prim(G,s)

A←{s}

B←V−{s}

for each v∈B　do

　if　v ∈ adj[s]　then

　　π[v] ←s

　　d[v] ←w((s,v))

　else

　　π[v] ←nil

　　d[v] ←∞

T←ϕ

while　|T| < |V|−1　do

　　p ←　B の中で d[v]が最小となる　v

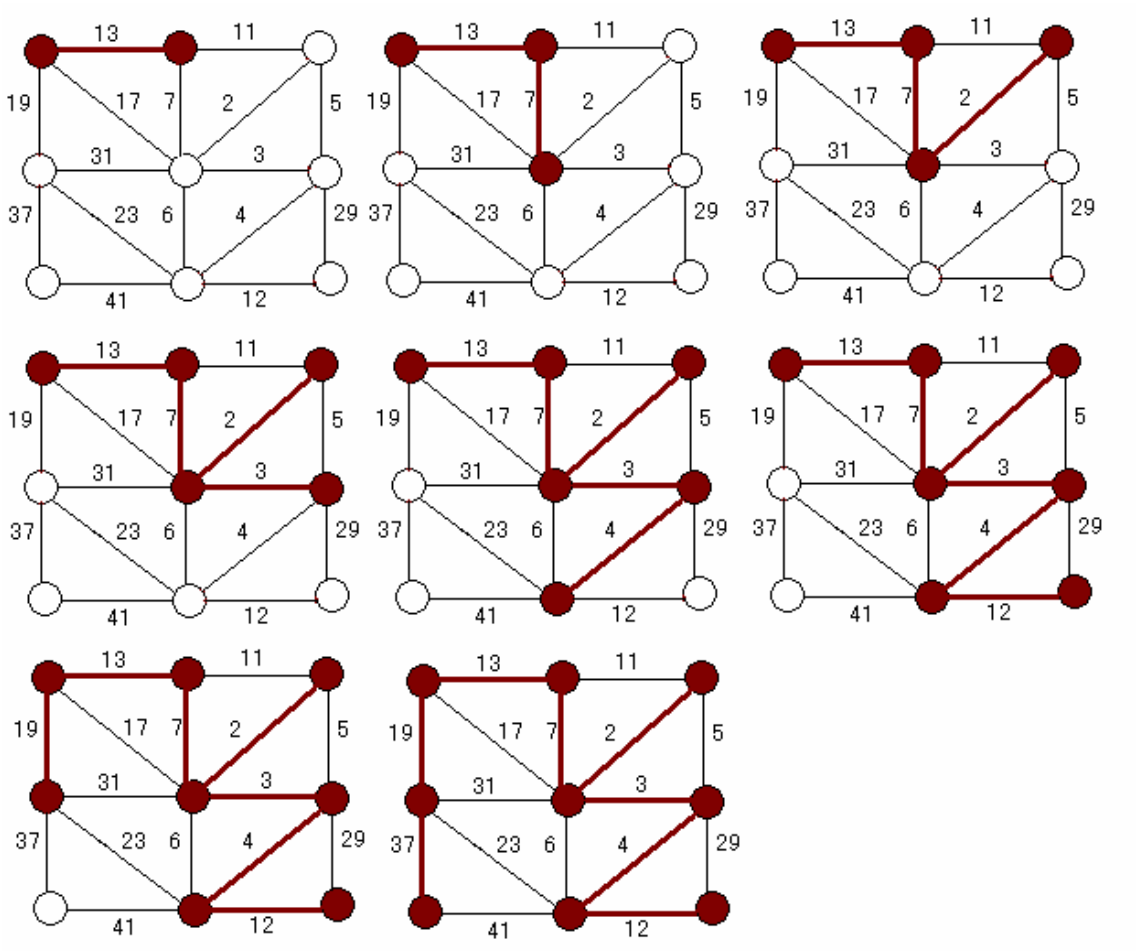
　　u ←　π[p]

　　A ←　A∪{ p }

B ← B－{ p }
T ← T∪{ (u, p) }
for each v ∈adj[p]∩B do
if d[v] > w((p,v)) then
d[v] ← w((p,v))
π[v] ← p

最初、G の節点全体を A={ s }と残りに分け、その間をむすんでいる枝の中から 1 番軽い枝(s,p)を選び、A={s,p}とする。これで、木に枝が１つ生えた。この操作を繰り返し、木を育てて生成木になったら終わる。

例２．Kruskal のときと同じグラフに MST_Prim を適用すると、次のように、最小木ができあがって行く。



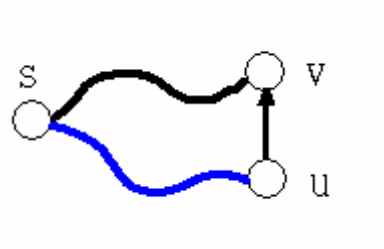
12.2　最短路問題

重み付きグラフ G=(V,E)の道 p ：e_1 e_2 ,⋯ ,e_t　の重み w(p)=Σ w(e_i)と定義する。節点 s から v への道の中で、最小の重みを持つ道を節点 s から v への最短路といい、その重みをδ(s,v)とかく。 s から到達不可能な節点 v に対しては　δ(s,v)=∞とする。重み付きグラフ G=(V,E)に関する最短路問題とは始点 s を固定して、s から各節点への最短路と最短路の重みを求める問題。(正確には単一始点最短路問題という) δ(s,v)について、次の性質が成り立つ。証明はいずれも簡単なので、練習問題とする。

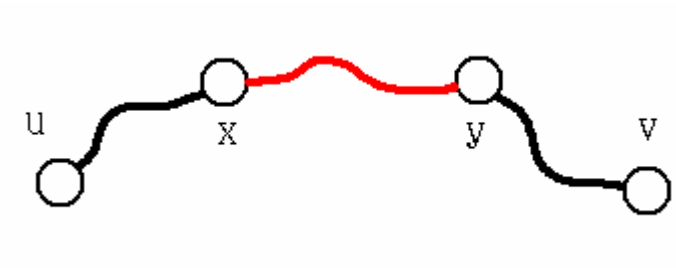
補題 12.3（三角不等式） G=(V,E)は重み w を持ったグラフで始点を s とする。任意の枝(u,v)について、

δ(s,v)≦　δ(s,u)+w(u,v)

が成り立つ。



補題 12.4　u から v への最短路を p とする。 p 上の任意の 2 点 x,y について、 p の x から y への部分 p[x,y] は x から y への最短路である。



【ダイクストラ・アルゴリズム】

さて、ダイクストラのアルゴリズムを紹介する。ダイクストラのアルゴリズムにおいては、**各枝の重みは 0 以上と仮定する。**このアルゴリズムでは s から各節点 v への道をみつけて、その重みを d[v]に記憶する。そして、さらに、重みの小さい道を発見したときには d[v]の値をそれに置き換える。その中心的操作がつぎの Relax(u,v,w)である。Q は優先度つきキューで、その中から、d[v]の最も小さい節点 v を取り出す操作を Extract_Min(Q)と書く。 π[v]には見つけた道の v の先行点を記憶することで、その道を記憶する。

Relax(u,v,w)

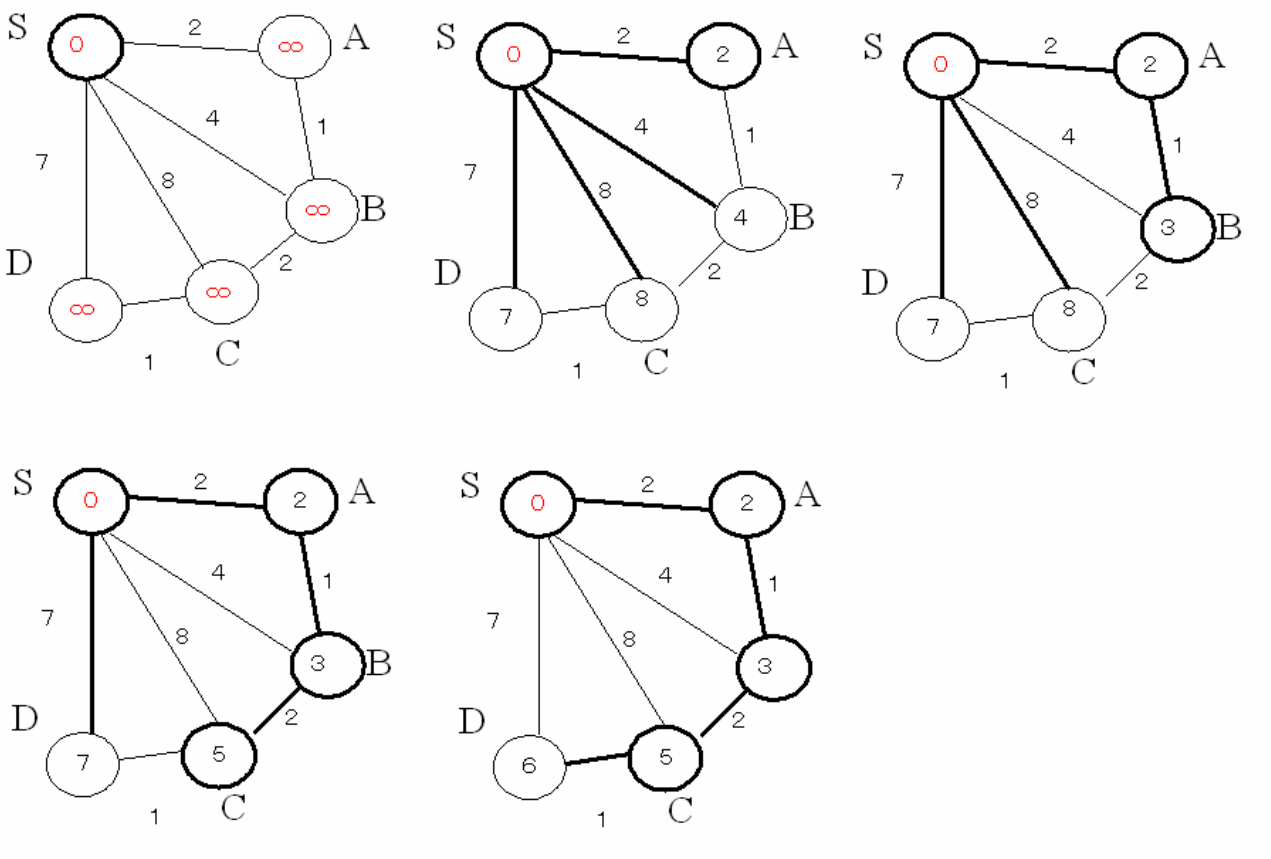
```
if d[v] > d[u]+w(u,v) then
       d[v] ← d[u]+w(u,v)
       π[v] ← u
```

Dijkstra(G,w,s)

```
for each v ∈V do
      d[v] ← ∞
      π[v] ← nil
d[s] ← 0
S ← φ
Q ← V
while Q ≠φ  do
    u ← Extract_Min(Q)
    S ← S∪{u}
   for each v ∈ adj[u] do
        Relax(u,v,w)
```

広さ優先探索のときに定義した部分グラフ $G_\pi=(V_\pi, E_\pi)$を考えると、V_π はｓから到達可能な節点の全体であり、$G_\pi=(V_\pi, E_\pi)$はｓを根とする木となる。これを最短路木という。ダイクストラのアルゴリズムの計算量は優先度つきキューにおけるExtract_Min(Q)の計算量に依存するが、例えばこれが2分木によるヒープで構成されている場合、1回につき、O(log|V|))であるから、ダイクストラのアルゴリズム全体の計算量はO(|E|log|V|)となる。

ダイクストラのアルゴリズム実行例

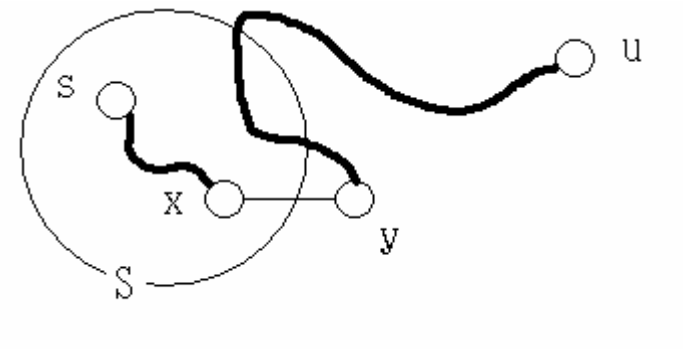


定理 12.4　G=(V,E)は重み w の重み付きグラフで、任意の e について、w(e)≧0 とする。s を始点として、Gにダイクストラのアルゴリズムを実行する。実行終了後、任意の節点 v について、d[v]=δ(s,v)が成り立つ。

証明：アルゴリズムの実行中において、いかなる時点でも　d[v]≧δ(s,v)　が成り立つ。実際、初期化（最初の６行）の直後に成り立つことは明らかである。それ以降、d[v]の値は Relax(u,v,w)でしか変化しないが、変化しても、その値は枝(u,v)を経由する s からの道の長さなので、d[v]≧δ(s,v)が成り立つ。ついでに注意しておくが、d[v]の値は変化しても小さくなるだけ、つまり、単調減少。また、いったん、d[v]=δ(s,v)が成り立つと、それ以降、この等号が最後まで成り立つ。

さて、定理 12.3 の証明に入る。u が集合Ｓに加えられたときに、d[v]=δ(s,v)が成り立つことを言えば十分である。(それ以降 d[v]の値は変化しない)背理法で示す。Ｓに加えられたときに、d[u]≠δ(s,u)であるような u があったとする。そのような節点のうち最初にＳに加えられたものを改めて u とする。d[s]= δ(s,s)=0 なので、u≠s である。よって、u がＳに加えられたときに、Ｓ≠φ。s から u への道がなければ、δ(s,u)＝∞なので、上に注意したように、d[u]≧δ(s,u)= ∞から d[u]＝δ(s,u)が成り立ち、u の取り方に反する。だから、s から u への道が存在する。そのような道の中で最短なものｐを

とる。さて、ｕがＳに加えられる直前を考える。



この道はＳの中からＳの外への道なので、この道の上の節点ｙで最初にＳに属さないものが存在する。ｙの一つ前の点をｘとする。ｘは ｕ より前にＳに加えられた節点なので、(u の取り方から)　d[x]=δ(s,x)　である。ｘがＳに加えられたときに、Relax(x,y,w)が実行されて、

d[y]≦d[x]+w(x,y)

　　=δ(s,x)+w(x,y)

　　=δ(s,ｙ)　　（ｐは最短路なので、その一部分であるｓからｙの部分はｓからｙへの最短路）

d[y]≧δ(s,ｙ)なので、d[y]＝δ(s,ｙ)

さて、**仮定「任意の e について、w(e)≧0　」より、δ(s,ｙ)≦δ(s,u)である。**

従って、

d[y]＝δ(s,ｙ)　≦δ(s,u)≦d[u]　　……(*)

さて、アルゴリズムの

u ← Extract_Min(Q)

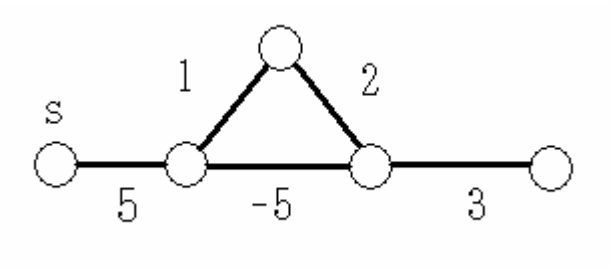
によってｕが優先度つきキューＱから取り出されたとき、ｙはまだＱの中にあるので、

d[u]≦ d[y]とわかる。従って、式(*)より、d[y]＝δ(s,ｙ)＝δ(s,u)＝d[u]　とくに、δ(s,u)＝d[u]となる。これはｕの取り方に反する。証明終わり。

【ベルマン-フォード・アルゴリズム】

負の値をもつ枝が存在するときを考える。もし、始点ｓから到達可能な範囲に負の値をもつ閉路が存在すれば、その閉路を回れば回るほど道の重みは小さくなる。つまり、最短路は存在しない。このような閉路があるかどうか判定し、存在しない場合には、始点から各点への最短路とその重みを求めるアルゴリズムがあると便利である。ベルマン-フォード（Bellman-Ford）アルゴリズムはそのようなアルゴリ

ズムである。




```
Bellman－Ford(G, w, s)
for each v ∈V do
        d[v] ← ∞
        π[v] ← nil
d[s] ← 0
for i=1 to |V|-1 do
    for each (u,v) ∈E  do
                Relax(u,v,w)
for each (u,v) ∈E  do
    if d[v] > d[u] + w(u,v)
       then return FALSE
return TRUE
```

定理 12.5　重みｗを持ったグラフ G=(V,E)にBellman－Ford(G, w, s)を実行したとする。もし、ｓから到達可能な範囲に負の重みを持った閉路がないならば、このアルゴリズムは TRUE を返し、すべてのｖに対して、d[v]=δ(s,v)となる。もし、ｓから到達可能な範囲に負の重みを持った閉路があるならば、このアルゴリズムはFALSE を返す。

証明：ｓから到達可能な範囲に負の重みを持った閉路がないとする。ｖがｓから到達不可能な場合、δ(s,v)=∞、ｄ[v]= ∞　なので成り立つ。ｖがｓから到達可能な場合、ｐ：$v_0 v_1 \cdots v_k$　をｓからｖへの最短路とする。ｓから到達可能な範囲に負の重みを持った閉路がないので、ｐは単純なものが取れる。つまり、v_0、v_1、…、v_k　はすべて異なる。よって、　$k \leqq |V|-1$。

一般に、ｐがｓからｖへの最短路で、ｖの先行点をu とする。また、d[u]=δ(s,u)とする。このとき、Relax(u,v,w)を実行すれば、d[v]=δ(s,v)となることに注意する。このことをｐ：$v_0 v_1 \cdots v_k$　についてｓに近いほうから順に適用する。Relax（v_0、v_1、ｗ）によって、ｄ[v_1]=δ（ｓ、v_1)、次に、

Relax（v_1、v_2、w）により、d[v_2]=δ（s、v_2）となっていく。よって、|V|－１回の反復ですべての点について、d[v]=δ(s,v)となる。これで、前半の証明が終わった。後半：sから到達可能な範囲に負の重みを持った閉路cがあるとする。c：$v_0 v_1 \cdots v_k$　（$v_0 = v_k$）とする。cの重みは負なので、

$$\Sigma w\ (v_{i-1}, v_i) < 0$$

である。もし、Bellman－Ford(G, w, s)がTRUEを返したとするならば、

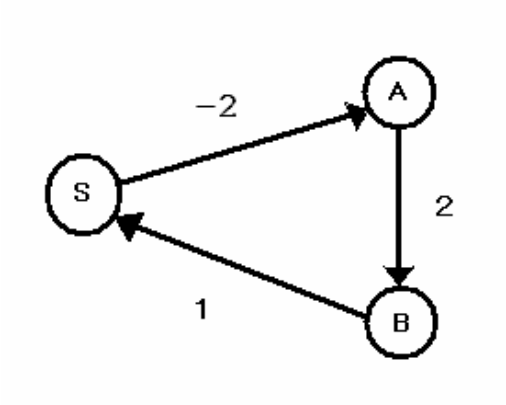
$$d[v_i] \leqq d[v_{i-1}] + w(v_{i-1}, v_i) \qquad (i=1,2, \cdots ,k-1)$$

がなりたっている。これらをすべて加えると、

$$\Sigma d[v_i] \leqq \Sigma d[v_{i-1}] + \Sigma w(v_{i-1}, v_i)$$

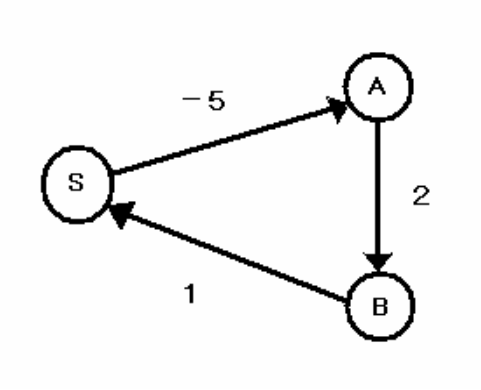
$v_0 = v_k$なので、$\Sigma d[v_i] = \Sigma d[v_{i-1}]$である。よって、$\Sigma w(v_{i-1}, v_i) \geqq 0$となり、cの重みは負であることに矛盾する。証明おわり。

例１　次のグラフにベルマン・フォードを実行してみる。



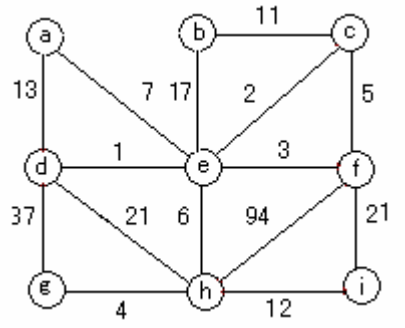
最初、d[A]=∞ ,　d[B]=∞, d[S]=0 であるが、relax(S,A)により、d[A]=-2, relax(A,B)により、d[B]=0, relax(B,S)のときに、d[S]=　0<d[B]＋１　なので、d[S]=0 のまま変更なし。2 回目のループで、d[A], d[B]、d[S]、すべて変化なし。アルゴリズムの後半の操作により、この場合、TRUE を返す

例２　次の負の重みの閉路を持つグラフにベルマン・フォードを実行してみる。



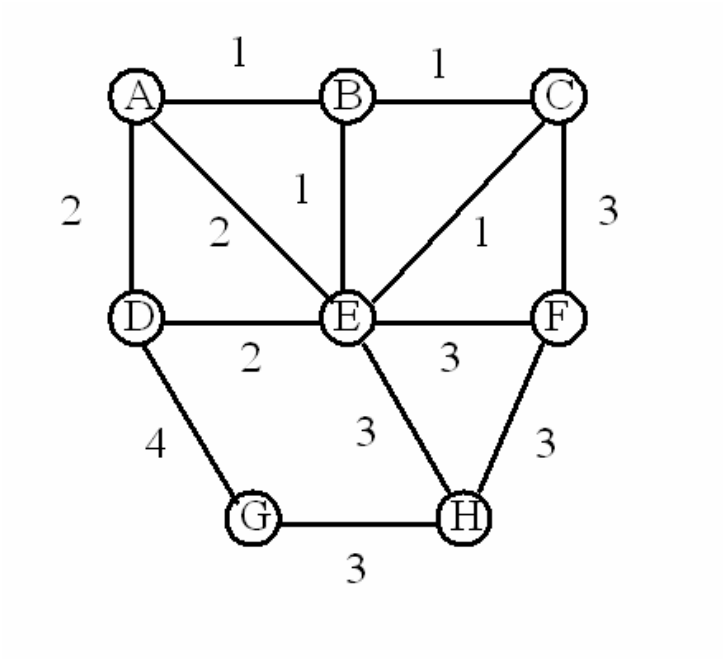
最初、d[A]= ∞ ，　d[B]=∞, d[S]=0 であるが、relax(S,A)により、d[A]=-5, relax(A,B)により、d[B]=-3, relax(B,S)により、d[S]=-2　となる。2 回目のループで、d[A]=-7, relax(A,B)により、d[B]=-5, relax(B,S)により、d[S]=-4　となる。それで、アルゴリズムの後半の操作により、この場合、FALSE を返す

練習問題 12.1　次の重み付きグラフの最小木を Kruskal の方法および Prim の方法のそれぞれで求めよ。その際の途中経過についても詳細に説明すること。なお、Prim の方法では出発点となる節点は a とする。



練習問題 12.2　12.1 の重み付きグラフについて、a から各節点への最短路とその重みをダイクストラのアルゴリズムにしたがって求めよ。また、その途中経過についても、d[ ]とπ[ ]の値の変化も含めて説明せよ。さらに、実行後にできる最短路木を図示せよ。

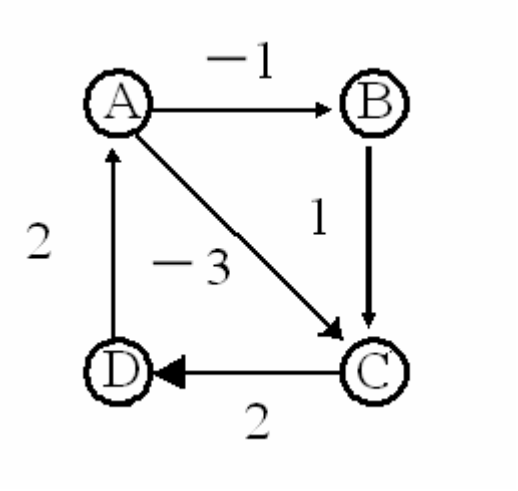
練習問題 12.3　次の連結無向グラフの最小木をすべて求めよ。



練習問題 12.4

連結無向グラフの１つの最小木をTとする。Tの枝の重みを昇順に$a_1, a_2, \cdots, a_m$とする。つまり、 $a_1 \leqq a_2 \leqq \cdots \leqq a_m$ とする。 この（$a_1, a_2, \cdots, a_m$）はすべての最小木に共通か？正しい場合には証明を与え、間違っている場合には反例を与えよ。

練習問題 12.5　次の重み付きグラフに対し、A を始点とするベルマン・フォード・アルゴリズムを実行するとする。実行終了時の各点ｖに対する　d[ｖ]，π[v]の値はどうなるか？



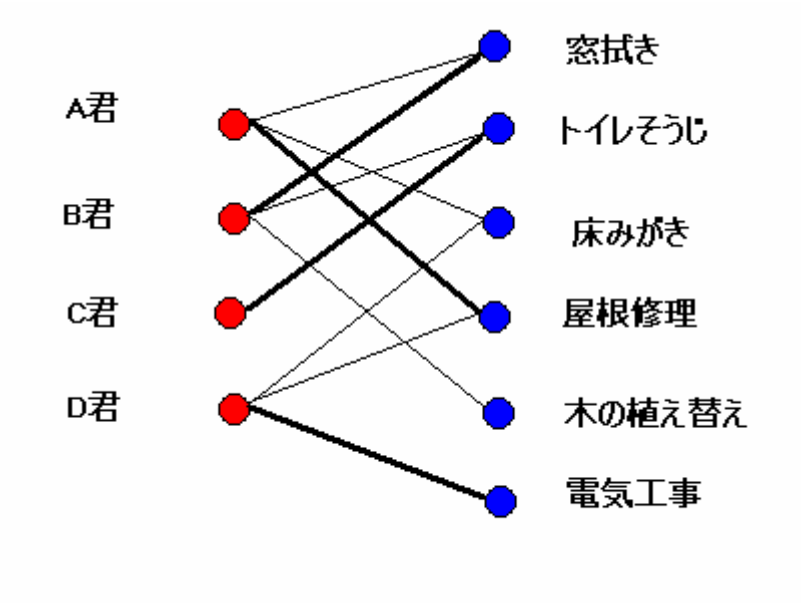
１３．マッチング

２部グラフG=（V、E)のマッチングについて説明する。 ２部グラフであるから、Vは２つに分割されていて、それをV_1, V_2と書く。$V_1 \cup V_2 = V$、$V_1 \cap V_2 = \phi$。

定義 13.1　E の部分集合 M がマッチング
⇔任意の v ∈V に対して、 v を端点として持つ M の枝は高々１つ。

定義 13.2　M に属する枝の端点となっている節点を M で被覆されている点という。

例. 仕事の割り当て
V_1が人の集合で、V_2がいくつかの仕事の集合。V_1の各人に可能な仕事を線で結ぶと２部グラフができる。そこから実際に誰が何をやるかを決めるのがマッチング。（１つの仕事を一人がやる）



最大マッチング問題
与えられた２部グラフに対して、｜M|が最大となるマッチング M を見つける問題。これを解決するために補充パスという概念を導入する。

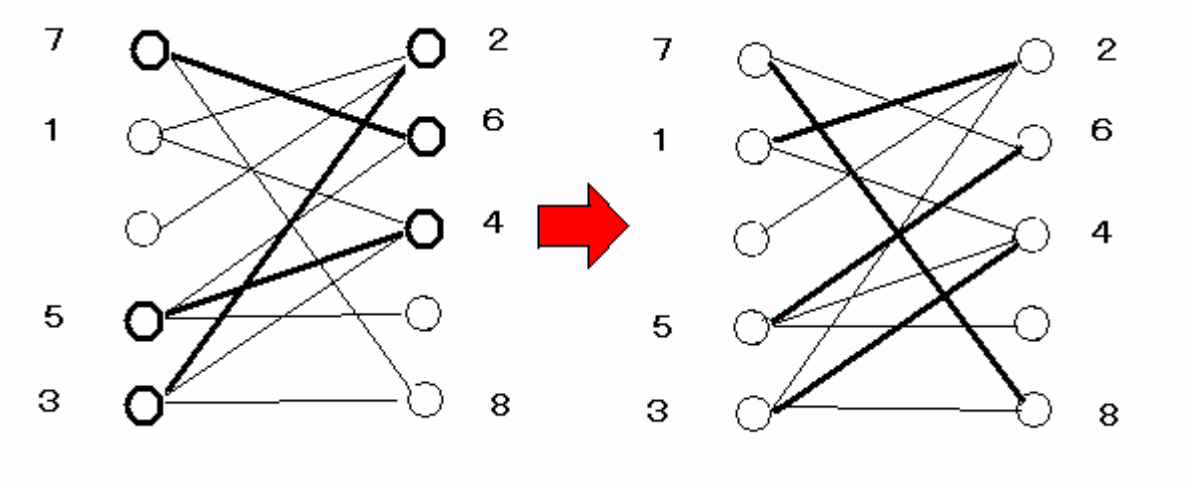
定義 13.3
２部グラフG=（V、E）　V= $V_1 \cup V_2$、$V_1 \cap V_2 = \phi$　とGのマッチングMに対して、Mで被覆されていない点からMで被覆されていない点への道 p ：$e_1, e_2, \cdots, e_{2k-1}$が$e_1 \notin M$、$e_2 \in M$、$e_3 \notin M$、…、$e_{2k-1} \notin M$ をみたすとき、 p は補充パスであるという。つまり、Mに属さない枝e_1から出発し、Mに属する枝、属さない枝を交互に通って、最後はMに属さない枝e_{2k-1}で終わる道のことを補充パスという。ここで、Mに属さない枝の方がMに属する枝より１つ多いことに注意する。補充パスが存在するとは限らないが、もし、見つかれば、新しいマッチングM´

$$M' = M \cup \{e_1,\ e_3, \cdots,\ e_{2k-1}\} - \{e_2,\ e_4, \cdots,\ e_{2k}\}$$

を考えれば、|M´|＝|M|＋１となる。始点と終点がMで被覆されていない点であったことと、それ以外の道の途中の節点が元々の M の枝の端点であったことより、これらはすべて異なり、従って、M´がマッチングになっていることがわかる。

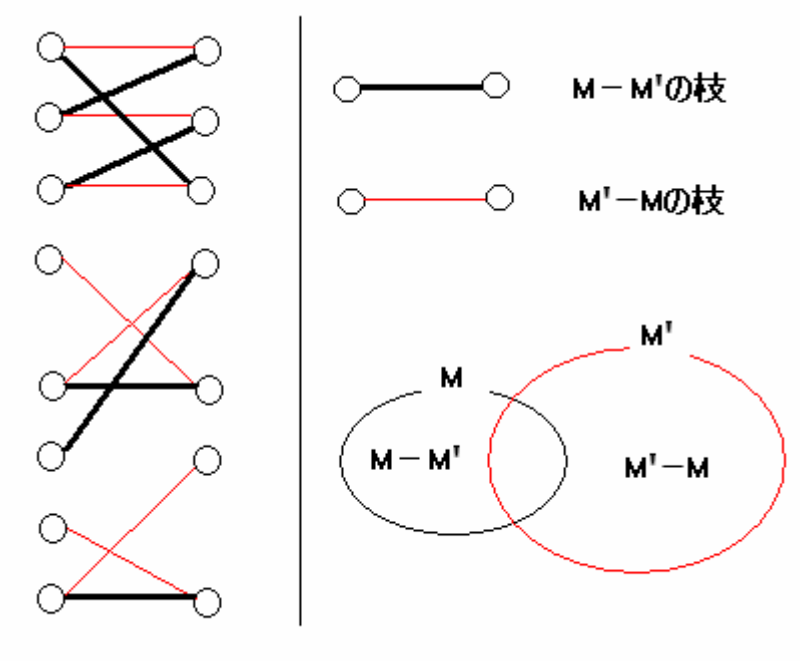
例

次の図で左側の１→２→３→４→５→６→７→８が補充パス。この補充パスにおいて太線と細線を入れ替えると右側の図になる。



定理 13.4　２部グラフ G=（V、E)のマッチングMについて、
Mが最大マッチング⇔Mに対する補充パスが存在しない。

証明：M に対する補充パスが存在すれば、上に述べたように、１つ要素の多いマッチング M´が作れるので、M は最大ではない。つまり、⇒は証明できている。そこで、M が最大マッチングでないときに、補充パスが存在することを示す。最大マッチングをM´とする。Mは最大マッチングでないので、|M´|＞|M|である。今、M´－M（≠φ）　に属する枝から出発し、M－M´の枝、その次に、M´－Mの枝というように交互に進めるだけ進む道全体を考える。これらの道がすべて、偶数個に枝で構成されているならば、|M´－M|＝|M－M´|となってしまう。|M´－M|＞|M－M´|なので、奇数個の枝で構成されているものが存在する。その道が補充パスである。



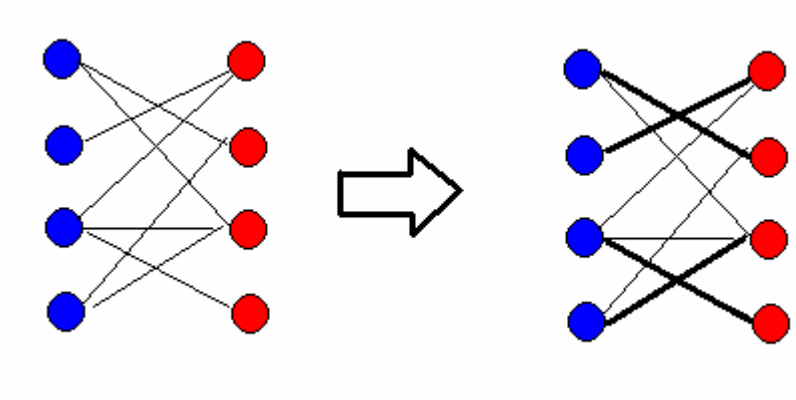
この定理により、最大マッチング問題は補充パスを見つけ、マッチングの個数を増やしていくことにより解決できる。

さて、2 部グラフG＝（V_1∪ V_2、E） のマッチングで、V_1のすべての点を被覆している（V_1からV_2への完全マッチングという）ものが存在するための条件を求める。例えば、仕事の割り当て問題でいうと、全員が何らかの仕事につくための条件ということになる。この問題は『結婚問題』として知られている。

【結婚問題】

m人の男性とｎ人の女性がいる。m人の男性はそれぞれ何人かの女性と知り合いである。すべての男性が知り合いの女性と結婚できるように組み合わせることができる条件を求めよ。

知り合い同士を線でつなぐと 2 部グラフができる。



V_1の部分集合Xに対して、adj(X)＝｛v　｜　vはXに属するある節点に隣接｝とおく。
2部グラフG＝（V_1∪ V_2、E)に対して、V_1からV_2への完全マッチングが存在するならば、

V_1の任意の部分集合Xに対して、｜adj(X)｜≧｜X｜　　…（＊）

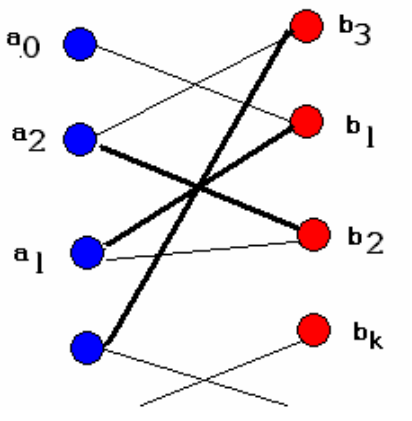
が成り立つ。実はこれが十分条件でもある。

定理 13.5（結婚定理　Hall 1935）
2部グラフG＝（V_1∪ V_2、E)に対して、V_1からV_2への完全マッチングが存在
⇔V_1の任意の部分集合Xに対して、｜adj(X)｜≧｜X｜

証明：⇐ を示せば良い。
Gのある最大マッチングMに対して、V_1のある節点a_0が被覆されていないとする。（＊）より、a_0には1点以上の隣接節点がある。その1つをb_1とする。b_1がMで被覆されていないなら、（a_0、b_1）が補充パスとなる。これはMの最大性に反する。よって、b_1はMで被覆されている。そのもう一方の端点をa_1とする。次の条件（＊＊）をみたす点列、a_0、b_1、a_1、b_2、…の存在が言えた。

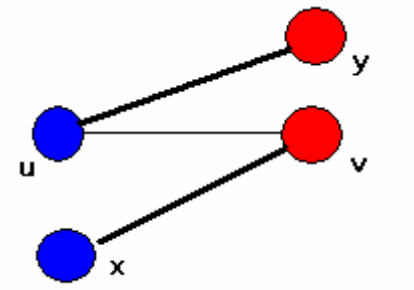
（＊＊）　a_0、b_1、a_1、b_2、…は異なる節点からなる点列で、a_i∈V_1、b_i∈V_2、
① a_0はMで被覆されていない。
② b_iはa_0、a_1、a_2、…、a_{i-1}のいずれかに隣接
③ （a_i、b_i）∈M

今、この（＊＊）をみたす点列の中で長さが最大なものをとってくる。｛a_0、a_1、a_2、…、a_{i-1}｝は条件（＊）より、いつもi 個の点と隣接しているので、、b_i≠b_1、b_2、…、b_{i-1} を条件②をみたすように選べる。したがって、この点列の最後の項はV_2の点である。 b_kをこの点列の最後の項とする。b_kは被覆されていない。もしされていたら、この点列はさらに拡大し、長さが最大なものをとったことに矛盾する。このように補充パスができて、これはMが最大マッチングであることに反する。よって、Mは完全マッチングである。



【安定結婚問題】

結婚問題にすこし条件をつける。何人かの男性と何人かの女性がいる。各人は結婚を希望する相手の順位をつけているとする。男性の集合をV_1、女性の集合V_2とする。結婚のマッチングに対して、つぎのような場合に$u \in V_1$と$v \in V_2$の『駆け落ち』がおこる。uが結婚した相手ｙより、ｖの方が順位が高く、ｖが結婚した相手ｘよりuのほうが順位が高い。

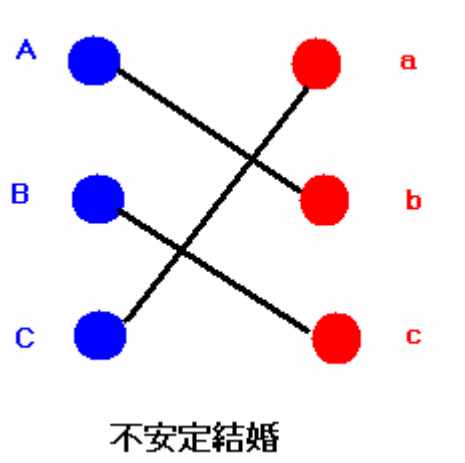


例、A,B,C の男性 3 人と a,b,c の女性 3 人に結婚する組を見つける。希望する相手の順位は

	1 位	2 位	3 位
A	a	b	c
B	b	a	c
C	a	c	b

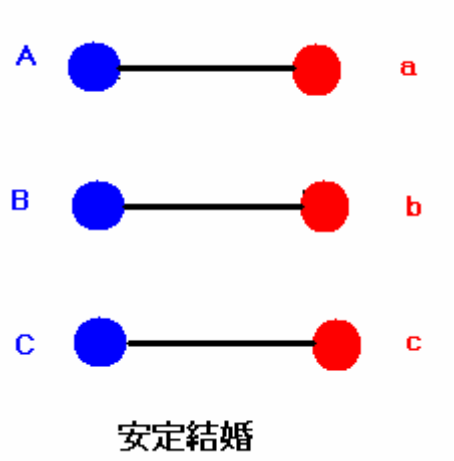
	1 位	2 位	3 位
a	B	A	C
b	C	B	A
c	A	C	B

とする。このとき、次のマッチングは不安定結婚である。



というのは（A， a）というペアを見てみると、Aにとっては現在の結婚相手 b より a の方が良いし、a にとっても現在の結婚相手Cより Aの方が良い。それではどう修正すれば、安定マッチングになるか

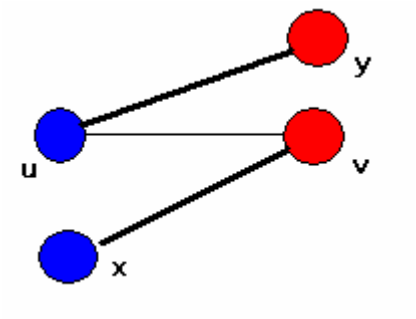
か？



これだと、A、Bとも第一希望の相手だし、Cはｃよりａの方が良いが、声をかけてもａはCを相手にしない。安定結婚となっている。

定義 13.6　一般に、2 部グラフ G=（V、E)に対して、各ｖ∈Vについて、希望リスト List[v]は順序＞をもっている（希望の大きい順に 1 列に並んでいる）とする。Gのマッチング M に対して、Mが安定とはつぎのような（u, v)∈Eが存在しないこと。

u のマッチング M による相手ｙはｖ＞ｙでしかも、ｖのマッチング M による相手ｘはｕ＞ｘ



定理 13.7　完全 2 部グラフ G=Km,m について、安定完全マッチングが存在する。

証明：次のアルゴリズムによる。π［v]にはｖのマッチングによる相手を記憶する。

Stable marriage(G)

for each ｖ ∈V_2

　π[v]←nil

```
for each u ∈V1
 propose(u)
```

propose(u)

```
 v←List[u]の先頭の要素
  refuse(v,u)
return
```

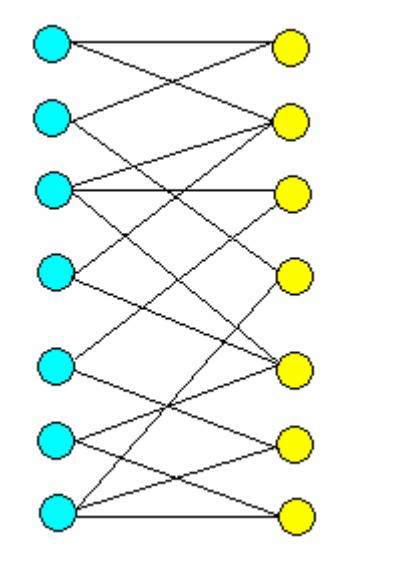
refuse(v,u)

```
 if π[v] ≠ nil  then
    x ←π[v]
     if x＞u then          {ｘ＞u はList[v]において、ｘの方がｕより順位が高いことを意味する}
        u を List[v]から削除   {v はｕを忘れ去る}
        v を List[u]から削除   {u はｖを諦める}
        propose(u)          {u は気を取り直して、次へアタック}
     else
       π[v]←u               {ｘと別れる}
       x を List[v]から削除    {v はｘを忘れ去る}
       v を List[x]から削除    {x はｖを諦める}
       propose(x)           {x は気を取り直して、次へアタック}

else                        {相手が決まっていないなら}
 π[v] ←u
 return
```

実行例： Stable marriage(G) をｐ４４の例に適用すると、どのようなマッチングができるか調べてみる。先ず、propose(A) で A のリストの先頭は a なので、refuse(a,A)が実行され、π[a]=A となる。次に、propose(B) で、refuse(b,B)が実行され、π[b]=B となる。次に、propose(C)が実行され、refuse(a,C)が実行される。a はすでに決まっている A の方が良いので、C からのプロポーズを拒否。つまり、a のリストから C を削除、C のリストから a を削除する。そして、propose(C)が実行される。このとき、リストの先頭はｃとなっていることに注意。つまり、refuse(c,C)が行われ、π[c]←C となる。これで終了。

練習問題 13.1　次の 2 部グラフの最大マッチングを 1 つ求めよ。



練習問題 13.2　次の 2 部グラフに完全マッチングが存在するかどうか調べよ。