Fluxo em Redes

Prof. Ricardo R. Santos

Problema da Árvore Geradora Mínima

- Grafo não-direcionado de *n* nós
- Problema de transporte em que existem suprimentos de *n-1* unidades de um produto em apenas 1 dos nós
- Existe demanda de 1 unidade do produto nos demais nós
- O custo de transporte em cada aresta *(i,j)* do grafo independe do volume transportado do produto na aresta
  - É zero se a aresta é utilizada no transporte ou c_{ij} (distância da aresta) se é utilizada
  - Adoção de variável binária para indicar 1 se há fluxo na aresta e 0 se não há fluxo na aresta

Problema da Árvore Geradora Mínima

- Modelo Matemático

$$\min \; f(x, y) = \sum_{(i,j)\in E} c_{ij} y_{ij}$$

$$\sum_{\{j:(1,j)\in E\}} x_{1j} = n-1$$

$$\sum_{\{i:(i,j)\in E\}} x_{ij} - \sum_{\{k:(j,k)\in E\}} x_{jk} = 1, j = 2,\ldots,n$$

$$x_{ij} \geq 0, (i, j) \in E$$

$$(n-1) y_{ij} \geq x_{ij}, (i, j) \in E$$

$$y_{ij} \leq x_{ij}, (i, j) \in E$$

$$y_{ij} \in \{0,1\}, (i, j) \in E$$

Problema da Árvore Geradora Mínima

- As variáveis:
  – x_{ij} é a quantidade transportada do produto do nó i para o nó j utilizando a aresta (i,j)
  – y_{ij} é a variável binária que indica se a aresta (i,j) é utilizada para transportar o produto do nó i para o nó j. Recebe valor 1 se a aresta (i,j) é utilizada e 0, caso contrário
  – c_{ij} é o “custo” incorrido caso a aresta (i,j) seja utilizada para se realizar este transporte do produto do nó i para o nó j

Problema da Árvore Geradora Mínima

- Procedimento guloso para escolher as arestas de menor comprimento
  - Algoritmo de Kruskal
- Algoritmo:
  - Dados:
    - G(N,E), em que N={1,2,...,n}
    - c(i,j) comprimento da aresta (i,j)
  - Saída:
    - ST, árvore geradora mínima
    - LST, comprimento da árvore geradora mínima

Problema da Árvore Geradora Mínima

- Algoritmo:
  - Passo 1:
    - LST=0
    - C={1} //escolha do nó 1 é arbitrária
    - C'=N-C
    - ST={}
  - Passo 2:
    - Enquanto (C' $\neq \emptyset$) faça
      - Selecione j $\in$ C' tal que c(k,j)=min {c(r,s) tal que (r,s)$\in$ E, r $\in$ C e s $\in$ C'}, isto é, o nó em C' mais próximo de qualquer nó em C. Seja k o nó em C mais próximo de j
      - C $\leftarrow$ C U {j}
      - C' $\leftarrow$ C' - {j}
      - LST $\leftarrow$ LST+c(k,j)
      - ST $\leftarrow$ ST U {(k,j)}

Fluxo em Redes - Definições

- Definição 1: Seja N um conjunto finito, cujos elementos são chamados ***nós*** (ou vértices) e E um conjunto de pares de nós, cujos elementos (i,j) são chamados ***arestas***. O par G=(N,E) é chamado ***grafo***. Uma ***rede*** é um grafo cujos nós e/ou arestas têm valores associados

- Definição 2: Um grafo G=(N,E) no qual as arestas são pares ordenados (subconjunto de NxN) é chamado ***grafo ordenado*** ou ***dígrafo***. Uma rede orientada é um grafo orientado cujos nós e/ou arcos têm valores associados

- Definição 3: O número de elementos de um conjunto X é chamado ***cardinalidade*** de X e denotamos por |X|

Fluxo em Redes - Definições

- Definição 4: Um ***caminho*** de um nó i_0 a um nó i_k é uma seqüência de arcos $C=\{(i_0,i_1),(i_1,i_2),(i_2,i_3),...,(i_{k-1},i_k)\}$, no qual o nó inicial de cada arco é o nó final do arco imediatamente anterior na seqüência, e $i_0,i_1,i_2,...,i_k$ são todos nós distintos

- Definição 5: Uma ***cadeia*** é uma estrutura similar à de um caminho, exceto que os arcos não precisam estar coerentemente orientados, ou seja, uma cadeia é uma seqüência de arcos de modo que cada arco tem exatamente um nó em comum com o arco imediatamente anterior na seqüência

- Definição 6: Um ***circuito*** é um caminho fechado, ou seja, é um caminho de um nó i_0 a um nó i_k, em que $i_k=i_0$. O correspondente ao circuito, no caso da cadeia, é denominado *ciclo,* ou seja, o ciclo é uma cadeia fechada

Fluxo em Redes - Definições

- Definição 7: Um grafo é ***fracamente conectado*** se existe pelo menos uma cadeia entre quaisquer dois de seus nós, e *fortemente conectado* se existe pelo menos um caminho de cada nó a todos os demais nós do grafo

- Definição 8: Uma ***árvore*** é um grafo conectado sem ciclos. Diz-se um grafo G'=(N',E') é um ***subgrafo*** de G=(N,E) se N'$\subseteq$ N e E'$\subseteq$ E com a condição de que, se (i,j) $\in$ E', então i e j também devem pertencer a N'. Uma árvore geradora de um grafo G é um subgrafo de G que é uma árvore e inclui todos os nós do grafo G

- Propriedade 1: Considere um grafo G=(N,E), com |N|=n (isto é, G tem *n* nós), e um subgrafo G'=(N,E') de G. As seguintes informações são equivalentes:
  – i) G'=(N,E') é uma árvore geradora de G
  – ii) |E'|=n-1 (isto é, G' tem n-1 arcos) e G' é conectado
  – iii) |E'|=n-1 e G' não tem ciclos

Problema do Caminho Mínimo

- Formulação Matemática
  - Problema do caminho mínimo do nó 1 para o nó n do grafo é um caso especial do problema de transporte onde deseja transportar, ao menor custo, uma unidade de um produto produzido no nó 1 para o nó n

Min $$f(x) = \sum_{i=1}^{n} \sum_{j \in S(i)} c_{ij} x_{ij}$$

$$\sum_{j \in S(1)} x_{1j} = 1$$

$$\sum_{i \in P(n)} x_{in} = 1$$

$$\sum_{i \in P(j)} x_{ij} = \sum_{k \in S(j)} x_{jk}, \; j = 2, \ldots, n-1$$

$$x_{ij} \geq 0, i = 1, \ldots, n \quad e \quad j = 1, \ldots, n$$

Problema do Caminho Mínimo

- Formulação Matemática
  - Em que:
    - S(j) é o conjunto dos nós sucessores de j
    - P(j) é o conjunto dos nós predecessores de j
    - x_{ij} é a quantidade transportada do produto da origem i para o destino j utilizando o arco (i,j)
    - c_{ij} é o “custo” incorrido por usar o arco (i,j)
- Para a resolução do problema do Caminho Mínimo existem algoritmos mais simples e eficientes que o método simplex: Algoritmo de Dijkstra, Algoritmo de Ford, Algoritmo de Floyd

Problema do Caminho Mínimo

- Algoritmo de Dijkstra
  - Dados:
    - G(N,E): grafo em que N={1,2,...,n}
    - 1: nó inicial do caminho
    - n: nó final do caminho
    - c(i,j): comprimento do arco (i,j) $\in$ E (hipótese: c(i,j)>=0)
  - Saída:
    - d(n): menor distância do nó 1 ao nó n
    - C: caminho mínimo entre o nó 1 e o nó n
  - **Encontra o menor caminho entre quaisquer dois nós da rede quando todos os arcos têm comprimentos não-negativos**

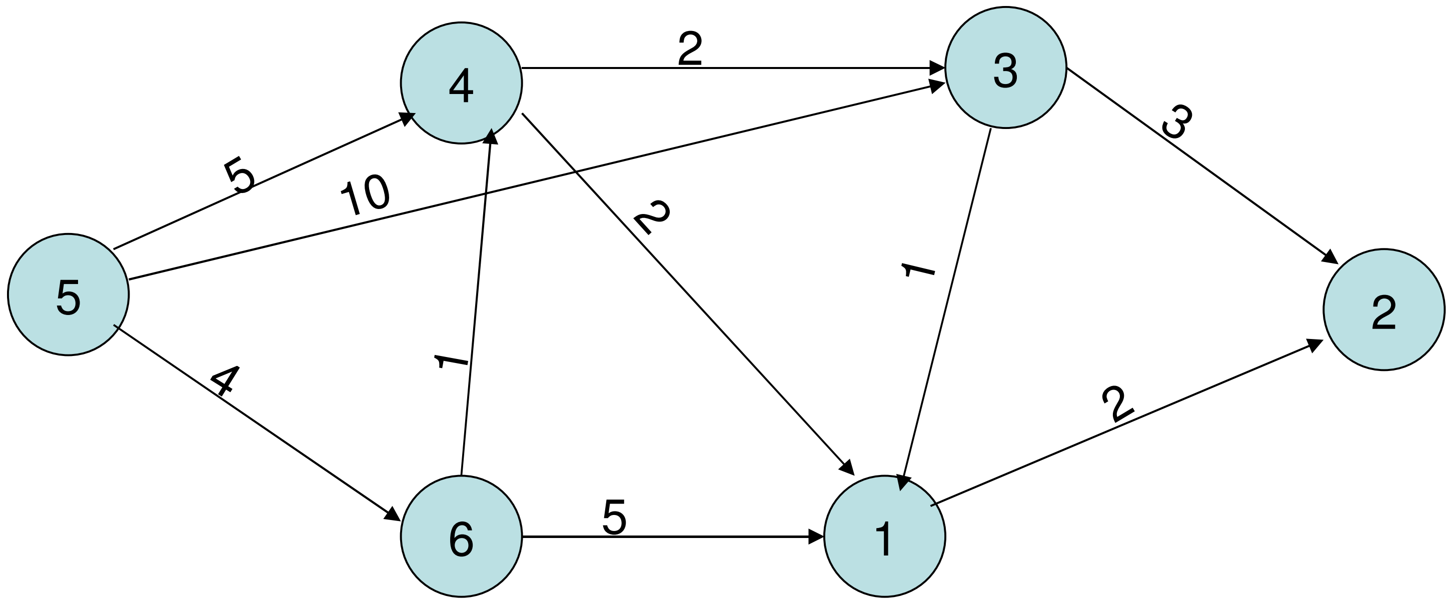
- O algoritmo de Dijkstra é descrito a seguir:
  - Separa-se os nós em rotulados (conjunto R) e não-rotulados (conjunto NR)
  - Nós rotulados são aqueles cuja ordenação já foi definida, ou seja, o nó mais próximo, o segundo nó mais próximos, etc.
  - Para recuperar um caminho mínimo até um nó *k*, guarda-se o nó anterior ao nó *k* no caminho (denotado por p(k), isto é, o caminho de 1 ao nó *k* é constituído do caminho de 1 ao nó p(k) e o arco (p(k),k).
  - Se p(k)==1, então o menor caminho que liga o nó 1 ao nó k é constituído tão somente do arco (1,k)

Problema do Caminho Mínimo

- Algoritmo de Dijkstra
  - Passo 1:
    - R={1}, NR={2,...,N}, d(1)=0, p(1)=0
    - Para i $\in$ NR,
      - d(i)=+∞
      - p(i)=n+1
      - a=1
  - Passo 2:
    - Para todo i $\in$ NR,
      - determine d(i)=minimo{d(i),d(a)+c(a,i)} e
      - faça p(i)=a, caso d(i)==d(a)+c(a,i)
    - Se d(i)==+∞ para todo i $\in$ NR, então pare //não existe caminho de 1 para outro nó
    - Senão, determine k $\in$ NR tal que d(k)=minimo{d(i),i $\in$ NR}.
      - NR=NR – {k}
      - R=R U {k}
      - a=k
  - Passo 3:
    - Se a==n então
      - Recupere o caminho mínimo C a partir dos valores armazenados em p(.), iniciando por k_1=p(n), em seguida, k_2=p(k_1), até que o nó 1 seja atingido. Senão, retorne ao Passo 2.

Problema do Caminho Mínimo

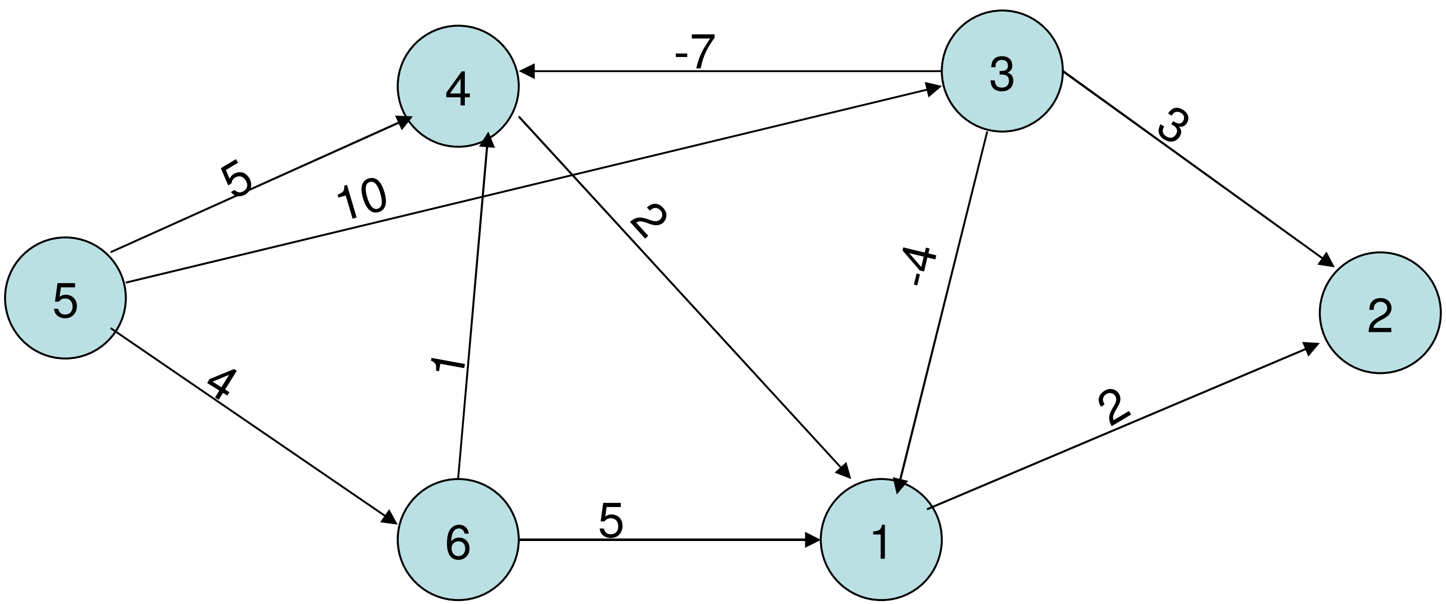
- Aplicar Alg. de Dijkstra sobre o grafo



- Encontrar menor caminho do nó 5 para o nó 2!
  - Mostre que C={(5,4),(4,1),(1,2)}

Problema do Caminho Mínimo

- Por que o Algoritmo de Dijkstra falha quando o grafo possui arestas negativas?



Problema do Caminho Mínimo

- Algoritmo de Ford
  - Encontra o caminho mais custo entre dois nós, mesmo que haja arcos com comprimentos negativos
  - É uma generalização do Alg. de Dijkstra
- Algoritmo
  - Dados:
    - G(N,E): grafo em que N={1,2,...,N}
    - 1: nó inicial do caminho
    - n: nó final do caminho
    - c(i,j): comprimento do arco (i,j), (i,j) $\in$ E
  - Saída:
    - d(n): menor distância do nó 1 ao nó n
    - C: caminho mais curto entre o nó 1 e o nó n

Problema do Caminho Mínimo

- Algoritmo
  - Passo 1: R={1}, NR={2,...,n}
    - d(1)=0, p(1)=0 e d(i)=+∞, p(i)=n+1, i ∈ NR
    - r(i)=0, i ∈ NR, r(1)=1
    - a=1
    - sinal=1 //sinal=0 quando todos os nós do grafo são rotulados
  - Passo 2:
    - Para todo v ∈ N
      - Calcule d(v)=min{d(v),d(a)+c(a,v)}
      - faça p(v)=a, caso d(v)=d(a)+c(a,v)
    - Se d(v)=+∞ para todo v ∈ NR então pare
    - Se para algum v ∈ R, d(v) decresceu de valor, então
    - R=R-{k} e NR=NR U {k}
    - Se NR≠{} então
    - a. Determine k tal que d(k)=minimo{d(v),v ∈ NR}
    - b. NR=NR – {k}, R=R U {k}, a=k, r(a)=r(a)+1
    - Senão, sinal=0
  - Passo 3:
    - Se sinal==0, recupere o caminho C a partir dos valores armazenados em p(.), iniciando por k_1=p(n), k_2=p(k_1), até que o nó 1 seja atingido e pare.
    - Se existe r(i)≥n, então pare: existe circuito no grafo com comprimento total negativo.
    - Caso contrário, retorne ao passo 2.

Problema do Fluxo Máximo

- Consiste em determinar o valor do maior fluxo possível que pode ser enviando de um nó a outro da rede
  - Exemplo: determinar a capacidade máxima de produção de um determinado produto
- Considera-se o nó 1 como nó origem (nó fonte) e o nó *n* como nó para o qual se deseja enviar o fluxo máximo (nó sorvedouro)
- Denota-se por *y* a quantidade do produto que está sendo enviado do nó 1 ao nó *n*, o modelo de otimização linear para este problema é

$$\text{Maximizar } y$$

$$\sum_{j \in S(1)} x_{1j} - \sum_{k \in P(1)} x_{k1} = y \qquad \text{(nó fonte 1)}$$

$$\sum_{j \in S(i)} x_{ij} - \sum_{k \in P(i)} x_{ki} = 0 \qquad \mathrm{i = 2, 3, \ldots, n\text{ -}1}$$

$$\sum_{j \in S(n)} x_{nj} - \sum_{k \in P(n)} x_{kn} = -y \qquad \text{(nó sorvedouro)}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{(para todo (i, j)} \in \text{E)}$$

Problema do Fluxo Máximo

- As equações do modelo anterior podem ser escritas com todas as variáveis no lado esquerdo

$$\sum_{j \in S(1)} x_{1j} - \sum_{k \in P(1)} x_{k1} - y = 0$$

$$\sum_{j \in S(i)} x_{ij} - \sum_{k \in P(i)} x_{ki} = 0$$

$$\sum_{j \in S(n)} x_{nj} - \sum_{k \in P(n)} x_{kn} + y = 0$$

- Onde pode-se concluir que a coluna da variável fluxo *y* pode ser vista como associada a um arco (n,1) chamado *arco de retorno*. O arco (n,1) passa a pertencer ao grafo e basta redefinir S(n)=S(n)+1

$$\text{Maximizar } x_{n1}$$

$$\sum_{j \in S(i)} x_{ij} - \sum_{k \in P(i)} x_{ki} = 0 \quad \mathrm{i} = 1, 2, 3, \ldots, \mathrm{n}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{para todo } (\mathrm{i}, \mathrm{j}) \in \mathrm{E}$$

Problema do Fluxo Máximo

- O algoritmo de Ford e Fulkerson
  - Todos os arcos do grafo têm capacidade $u_{ij}>0$.
    - Arcos com $u_{ij}=0$ obrigam eu $x_{ij}=0$ para toda solução factível
  - Supor que tem disponível alguma solução factível com um fluxo x^* indo do nó 1 ao n (exemplo, $x^*=0$). Para melhorar essa solução x^* é preciso encontrar algum caminho no grafo do nó 1 ao nó *n* por onde o fluxo atual pode ser incrementado
  - Pode-se determinar os arcos em dois tipos:
    - Classe A: arcos que podem ter fluxo aumentado (isto é, $x^*_{ij}<u_{ij}$)
    - Classe B: arcos que podem ter fluxo diminuído (isto é, $0<x^*_{ij}$)

Problema do Fluxo Máximo

- O algoritmo de Ford e Fulkerson
  - Pode-se construir um grafo auxiliar com os mesmos nós do grafo original definidos por:
    - Para cada arco (i,j) da classe A, há um arco (i,j) no grafo auxiliar com capacidade $(u_{ij}-x^*_{ij})$
    - Para cada arco (i,j) da classe B, há um arco (j,i) no grafo auxiliar com capacidade x^*_{ij}
      - Isto quer dizer que o fluxo neste arco pode ser diminuído de até x^*_{ij}
  - Não é preciso construir explicitamente o grafo auxiliar. Nesse caso, busca-se no grafo original uma cadeia do nó 1 ao nó n
  - Essa busca pode ser feita rotulando-se nós de maneira sucessiva, a partir do nó 1. Um nó *j* é rotulado se existir alguma nó *i* já rotulado e existir um arco (i,j), com a condição de que o fluxo no arco (i,j) pode ainda ser aumentado, ou se existir um arco (j,i) com a condição de que o fluxo no arco (j, i) possa ser diminuído
  - Observe que rotulando sucessivamente a partir do nó 1 se, em algum momento, rotulamos o nó *n*, teremos achado uma cadeia do nó 1 ao nó *n* por onde o fluxo pode ser aumentado

Problema do Fluxo Máximo

- O algoritmo de Ford e Fulkerson
  - Para recuperar a cadeia descoberta, utilizamos um vetor p, onde p(j) armazena *i* indicando que a partir do nó *i* rotulamos o nó *j*
  - Utilizamos sinal(j) para indicar se o arco (i, j) será aumentado, sinal(j)=+1, ou diminuído, sinal(j)=-1
  - Para arcos com capacidades inteiras, o algoritmo executa em ***O(Ef),*** onde E=Nº de arestas e f=fluxo máximo

- Passos do algoritmo de Ford e Fulkerson
  - Dados:
    - G(N,E), onde N={1, 2, ..., n}
    - u(i, j) capacidade máxima do arco (i, j) (u(i,j)>0)
    - 1 nó fonte
    - n nó sorvedouro
  - Saída:
    - Fluxo máximo *y* do nó 1 ao nó *n*

Problema do Fluxo Máximo

- Passos do algoritmo de Ford e Fulkerson
  - Passo 1: Início
    - y=0 (fluxo inicial x*=0)
    - v+(i, j)=u(i, j) para todos os arcos (i, j) $\in$ E
    - v-(i, j)=0 para todos os arcos (i, j) $\in$ E
    - R={n}: conjunto de nós rotulados
  - Passo 2: Enquanto n $\in$ R faça
    - R=$\emptyset$ (nenhum nó está rotulado inicialmente)
    - para i $\in$ N {p(i)=0 }
    - R={1} e Lista={1}
    - Enquanto (Lista $\neq\emptyset$) ou (n $\notin$ R) faça
      - Escolha i $\in$ Lista e faça Lista=Lista – {i}
      - Para todo arco (i, j) com (v+(i, j)>0) e (j $\notin$ R), faça
        - » p(j)=i
        - » sinal(j)=+1
        - » R=R U {j}
        - » Lista=Lista U {j}
      - Para todo arco (j, i) com (v-(i, j)>0) e (j $\notin$ R), faça
        - » p(j)=i
        - » sinal(j)=-1
        - » R=R U {j}
        - » Lista=Lista U {j}
    - Fim_Enquanto
    - Se (n $\in$ R), execute aumente_fluxo_atualize_fluxo
  - Fim_Enquanto

Problema do Fluxo Máximo

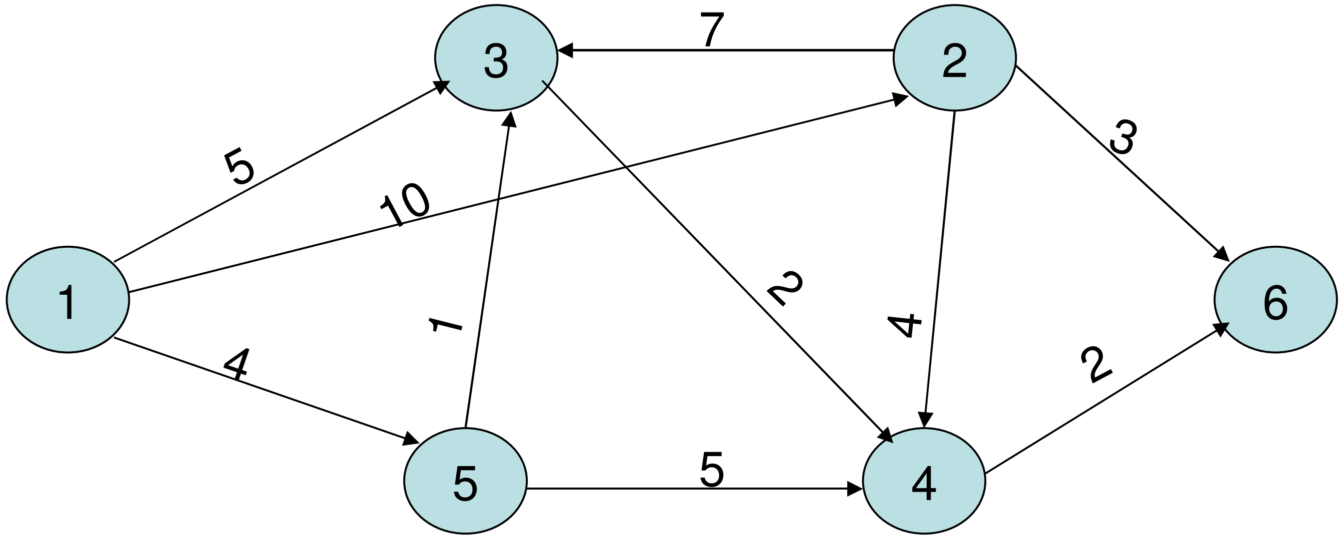
- Passos do algoritmo aumente_fluxo_atualize_fluxo
  - Dados:
    - G(N, E)
    - v+(i, j): capacidade máxima de aumento de fluxo no arco (i, j)
    - V-(i, j): capacidade máxima de diminuição de fluxo no arco (i, j)
    - 1: nó fonte
    - n: nó sorvedouro
    - y: fluxo atual
    - p(i): nó a partir do qual o nó i foi rotulado
    - sinal(i): se igual a +1, indica que o arco que deve ser recuperado para determinar a cadeia C será da forma (p(i), i); se igual a -1, indica que o arco que deve ser recuperado para determinar a cadeia C será da forma (i, p(i))
  - Saída:
    - Fluxo aumentado *y* do nó 1 ao nó *n*
    - v+(i, j): valor atualizado da capacidade máxima de aumento de fluxo no arco (i, j)
    - v-(i, j): valor atualizado da capacidade máxima de diminuição de fluxo no arco (i, j)

Problema do Fluxo Máximo

- Passos do algoritmo aumente_fluxo_atualize_fluxo
  - Passo 1:
    - r=n
    - C=∅
    - δ=+∞
    - Enquanto r ≠ 1, faça
      - Se sinal(r)==+1 então
        - » C=C U {(p(r),r)}
        - » δ=minimo{δ,v+(p(r),r)}
      - Senão
        - » C=C U {(r, p(r))}
        - » δ=minimo{δ,v-(r, p(r))}
      - r=p(r)
    - Fim_Enquanto
  - Passo 2: y=y+ δ
    - Para todo (i, j) $\in$ C
      - Se sinal(j)==+1 entao
        - » v+(i, j)=v+(i, j)- δ
        - » v-(i, j)=v-(i, j)+ δ
      - Se sinal(j)==-1 entao
        - » v-(i, j)=v-(i, j)- δ
        - » v+(i, j)=v+(i, j)+ δ

Problema do Fluxo Máximo

- Aplicar algoritmo de Ford e Fulkerson no seguinte grafo



Problema do Fluxo Máximo

- Exemplo com o algoritmo de Ford e Fulkerson
  - Passo 1: Início
    - y=0
    - v+(1,5)=4, v+(1,2)=10, v+(1,3)=5, v+(5,3)=1, v+(5,4)=5, v+(2,3)=7, v+(2,4)=4, v+(2,6)=3, v+(3,4)=2, v+(4,6)=2, v-(1,5)=0, v-(1,2)=0, v-(1,3)=0, v-(5,3)=0, v-(5,4)=0, v-(2,3)=0, v-(2,4)=0, v-(2,6)=0, v-(3,4)=0, v-(4,6)=0
    - R={6}
  - Passo 2:
    - R=∅
    - p(1)=0, p(5)=0, p(2)=0, p(3)=0, p(4)=0, p(6)=0
    - R={1} e Lista={1}
    - Lista=∅
      - p(5)=1, sinal(5)=+1, R={1,5}, Lista={5}
      - p(2)=1, sinal(2)=+1, R={1,5,2}, Lista={5,2}
      - p(3)=1, sinal(3)=+1, R={1,5,2,3}, Lista={5,2,3}
    - Lista={2,3}
      - p(4)=5, sinal(4)=+1, R={1,5,2,3,4}, Lista={2,3, 4}
    - Lista={3,4}
      - p(6)=2, sinal(6)=+1, R={1,5,2,3,4,6}, Lista={3, 4, 6}

Problema do Fluxo Máximo

- Exemplo com o algoritmo de Ford e Fulkerson
- Procedimento aumente_fluxo_atualize_fluxo
  - Passo 1: Início
    - p(6)=2, sinal(6)=+1
    - p(2)=1, sinal(2)=+1
    - C={(2,6), (1,2)}
  - Passo 2:
    - $\delta=$min{v+(2,6),v+(1,2)}=3
    - y=0+3=3
    - v+(1,2)=10-3=7
    - v-(1,2)=0+3=3
    - v+(2,6)=3-3=0
    - v-(2,6)=0+3=3

Problema do Fluxo Máximo

- Exemplo com o algoritmo de Ford e Fulkerson
    - R={1,5,2,3,6,4}
    - R=∅
    - p(1)=0, p(5)=0, p(2)=0, p(3)=0, p(4)=0, p(6)=0
    - R={1} e Lista={1}
    - Lista=∅
        - p(5)=1, sinal(5)=+1, R={1,5}, Lista={5}
        - p(2)=1, sinal(2)=+1, R={1,5,2}, Lista={5,2}
        - p(3)=1, sinal(3)=+1, R={1,5,2,3}, Lista={5,2,3}
    - Lista={2,3}
        - p(4)=5, sinal(4)=+1, R={1,5,2,3,4}, Lista={2,3, 4}
    - Lista={2,3}
        - p(6)=4, sinal(6)=+1, R={1,5,2,3,4,6}, Lista={2,3, 6}

Problema do Fluxo Máximo

- Exemplo com o algoritmo de Ford e Fulkerson
- Procedimento aumente_fluxo_atualize_fluxo
  - Passo 1: Início
    - p(6)=4, sinal(6)=+1
    - p(4)=5, sinal(4)=+1
    - p(5)=1, sinal(5)=+1
    - C={(1,5), (4,6), (5,4)}
  - Passo 2:
    - $\delta =$min{v+(1,5),v+(5,4),v+(4,6)}=2
    - y=3+2=5
    - v+(1,5)=4-2=2
    - v-(1,5)=0+2=2
    - v+(5,4)=5-2=3
    - v-(5,4)=0+2=2
    - v+(4,6)=2-2=0
    - v-(4,6)=0+2=2

Problema do Fluxo Máximo

- Exemplo com o algoritmo de Ford e Fulkerson
    - R={1,5,2,3,6,4}
    - R=∅
    - p(1)=0, p(5)=0, p(2)=0, p(3)=0, p(4)=0, p(6)=0
    - R={1} e Lista={1}
    - Lista=∅
        - p(5)=1, sinal(5)=+1, R={1,5}, Lista={5}
        - p(2)=1, sinal(2)=+1, R={1,5,2}, Lista={5,2}
        - p(3)=1, sinal(3)=+1, R={1,5,2,3}, Lista={5,2,3}
    - Lista={2,3}
        - p(4)=5, sinal(4)=+1, R={1,5,2,3,4}, Lista={2,3, 4}
    - Lista={3,4}
    - Lista={4}
    - Lista=∅

– **O fluxo máximo do nó 1 ao nó 6 é y=5**