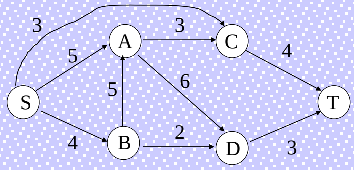
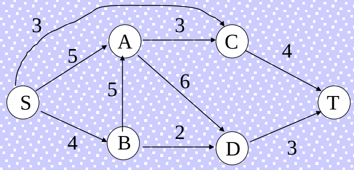
UM PROBLEMA DE FLUXO

- Imagine um sistema de fornecimento de água como o da Fig. baixa.
- Cada arco representa uma tubulação e o numero posicionado acima de cada arco representa a capacidade dessa tubulação em litros por minuto;
- Os nós representam pontos de união das tubulações e de transferência da água de uma tubulação a outra;



- Dois nós, *S* e *T*, são designados como uma fonte de água e um usuário de água, respectivamente. Isso significa que a água que se origina em *S* deve ser canalizada pela tubulação até *T*. A água só poderá fluir através de uma tubulação em uma direção (podem ser usadas válvulas de pressão para evitar o refluxo da água), e não existem tubulações entrando em *S* ou saindo de *T*.



Objetivo:

Gostaríamos de maximizar a quantidade de água fluindo da fonte para o usuário. O fator limitante do sistema inteiro é a capacidade de canalização.

Observação:

Existem outros problemas do mundo real semelhantes a esses. O sistema poderia ser uma rede elétrica, um sistema ferroviário, uma rede de comunicações ou qualquer outro sistema de distribuição no qual se queira aumentar a quantidade de um item sendo liberado de um ponto para outro.

Formulação do Problema

- Defina uma função de capacidade, *c*(*a*, *b*), onde *a* e *b* sejam nós, como segue, se *adjacent*(*a*, *b*) for TRUE, o valor de *c*(*a*, *b*) será a capacidade da canalização de *a* até *b*. Se não existir nenhuma tubulação de *a* até *b*, *c*(*a*, *b*) = 0.

Formulação do Problema (cont.)

- Defina uma função de fluxo, *f*(*a*, *b*), onde *a* e *b* sejam nós, como a quantidade de água fluirá através de cada tubulação, e *f*(*a*, *b*) = 0 se *adjacent*(*a*, *b*) = FALSE. Obviamente, $0 \leq f(a, b) \leq c(a, b)$, para todos nós *a* e *b*, porque uma tubulação não pode canalizar uma quantidade de água superior à sua capacidade.

Formulação do Problema (cont.)

- Considere *v* a quantidade de água que flui através do sistema de *S* até *T*. Sendo assim, a quantidade de água saindo de *S* através de todas as tubulações é igual à quantidade de água entrando em *T* através de todas as tubulações, e esse dois valores são iguais a *v* . Isto pode ser declarado pela igualdade

$$\sum_{x \in nós} f(S, x) = v = \sum_{x \in nós} f(x, T)$$

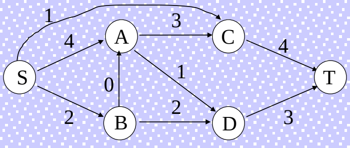
Formulação do Problema (cont.)

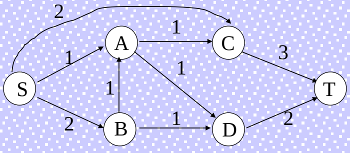
• Nenhum nó, exceto S, poderá produzir água e nenhum nó, exceto T, poderá consumir água. Sendo assim, a quantidade de água saindo de qualquer nó diferente de *S* ou *T* será igual à quantidade de água entrando neste nó. Isto pode ser declarado por

$$\sum_{y \in nós} f(x, y) = \sum_{z \in nós} f(z, x)$$

Para todos nós x != *S*, *T*.

• Podem existir varias funções de fluxo para determinado grafo e função de capacidade. As seguintes figuras ilustram duas possíveis funções de fluxo para o grafo anterior.





• Agora, o objetivo é determinar uma função de fluxo que maximize o valor de v, a quantidade de água canalizada de *S* para *T*. Essa função de fluxo é chamada **ótima**.

• **Idéia Básica** Uma função de fluxo pode ser obtida definido-se *f*(*a*, *b*) = 0 para todos os nós *a* e *b*. Evidentemente, essa função de fluxo é a menos ótima porque nenhuma água fluirá de *S* para *T*. Dada uma função de fluxo, ela poderá ser melhorada para que o fluxo de *S* para *T* seja aumentado. Então, A estratégia para produzir uma função de fluxo ótima é começar com uma função de fluxo 0 e melhorá-la sucessivas vezes até que uma função de fluxo ótima seja produzida.

Observação:

Entretanto, cada versão melhorada deverá atender a todas as condições impostas para uma função de fluxo valida. Em particular, se o fluxo entrando em qualquer nó (exceto *S* ou *T*) for aumentado ou diminuído, o fluxo saindo desse nó deverá ser aumentado ou diminuído concomitantemente.

Melhorando uma Função de Fluxo

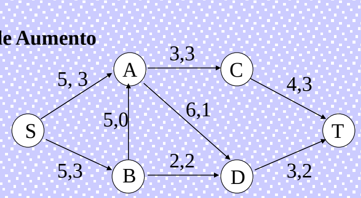
1) Aumento

Acha um caminho $S = x_1, x_2, ..., x_n = T$ de *S* a *T*, tal que o fluxo ao longo de cada arco no caminho seja estritamente inferior à capacidade, i.e., $f(x_{k-1}, x_k) < c(x_{k-1}, x_k)$ para todo *k* entre 1 e *n*. O fluxo pode ser aumentado em cada arco do caminho pelo valor mínimo de $c(x_{k-1}, x_k) - f(x_{k-1}, x_k)$ para todo *k* entre 1 e *n*.

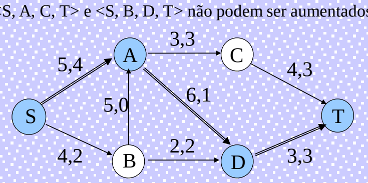
2) Redução

Se um nó *x* for adjacente a algum nó *y*, i.e., existe um arco <*y*, *x*>, a quantidade de água emanando de *y* na direção de *T* poderá ser aumentada, reduzindo-se o fluxo de *y* para *x*.

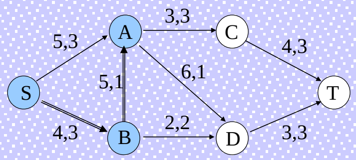
Exemplo de Aumento



Caminhos <S, A, C, T> e <S, B, D, T> não podem ser aumentados. Porquê?

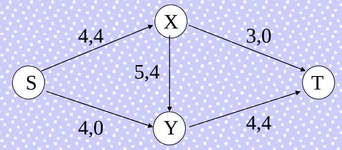


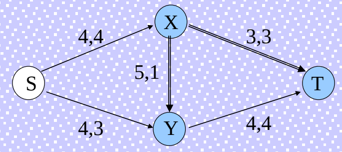
Aumento do caminho <S, A, D, T>.
O fluxo total de S para T foi aumentado de 5 a 6



Este aumento não melhora a função de fluxo

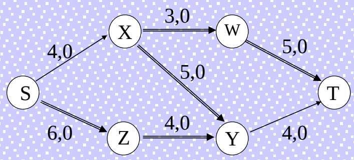
Exemplo de Redução





Fundamento do Algoritmo

- Defina um **semicaminho** de *S* para *T* como uma seqüência de nós $S = x_1, x_2, ..., x_n = T$ tal que para todo *i* entre 1 e *n*, ou $<x_{i-1}, x_i>$ ou $<x_i, x_{i-1}>$ seja um arco.



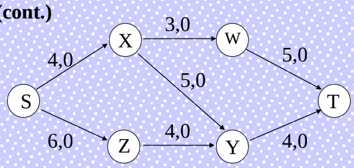
Fundamento do Algoritmo (cont.)

- Considere um semicaminho parcial descoberto a partir de de *S*. Se o ultimo nó num semicaminho descoberto a partir de *S* for *a*, o algoritmo considerara a extensão dele para qualquer outro nó *b* de modo que <*a*, *b*> ou <*b*, *a*> seja um arco. O semicaminho parcial só será estendido até *b* se a extensão puder ser feita de tal forma que o fluxo de entrada para *b* possa ser aumentado.
- Esse processo continua até que um semicaminho parcial a partir de *S* esteja finalizado, estendendo-o para *T*. Em seguida, o algoritmo retrocede ao longo do semicaminho, ajustando todos os fluxos até que *S* seja alcançado.

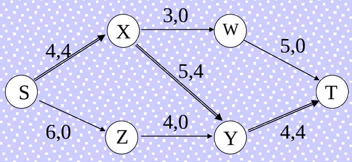
Fundamento do Algoritmo (cont.)

- O processo inteiro é então, repetido numa tentativa de descobrir ainda outro semicaminho de S para T. Quando nenhum semicaminho parcial puder ser estendido com sucesso, o fluxo não poderá ser aumentado e o fluxo existente será considerado ótimo.

Exemplo (cont.)



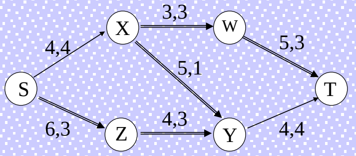
1ª semicaminho
SXYT



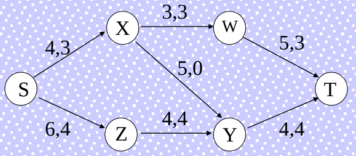
Exemplo (cont.)

2ª semicaminho
SZYXWT

3ª SZYX
sem sucesso
o processo termina



Outro fluxo ótimo



Algoritmo de Ford-Fulkerson

```
1   inicializa a função de fluxo com 0 em cada arco;
2   canimprove = TRUE;

3   do {
4       tentativa de achar um semicaminho de S p/ T que
        aumenta o fluxo p/ T por x > 0;

5       if (impossível encontrar semicaminho)
            canimprove = FALSE;
6       else
            aumente o fluxo para cada nó (exceto S) no
            semicaminho por x
7   } while (canimprove = TRUE);
```

Obs.: O coração do algoritmo reside na linha 4

Algoritmo para calcular fluxo máximo a partir de S que as tubulações podem suportar (linha 4)

onpath[node] - sinalizador que indique se *node* encontra-se ou não em algum semicaminho;

endpath[node] - sinalizador que indique se *node* está ou não na extremidade do semicaminho parcial;

precede[node] - aponta p/ o nó que precede *node* em seu semicaminho;

forward[node] - tem o valor TRUE se e somente se o arco estiver na direção de precede[node] para *node*;

Algoritmo para calcular fluxo máximo a partir de S que as tubulações podem suportar (linha 4) (cont.)

improve[node] - indica a quantidade pela qual o fluxo para *node* pode ser aumentado ao longo de seu semicaminho;

c[a][b] - a capacidade de canalização de *a* para *b*;

f[a][b] - o fluxo atual de *a* para *b*.

```
define endpath[node], onpath[node] com FALSE p/ todos os nós
endpath[S] = TRUE;
onpath[S] = TRUE;
improve[S] = soma de c[S][node] sobre todos os nós node;
while ((onpath[T] == FALSE) && (existir um nó nd tal que endpath[nd] = TRUE)) {
    endpath[nd] = FALSE;
    while (existir um nó i tal que (onpath[i] = FALSE) && (adjacent(nd, i) = TRUE)
            && (f[nd][i] < c[nd][i])) {
        /* o fluxo de nd p/ i pode ser aumentado. Coloque i no semicaminho */
        onpath[i] = TRUE; endpath[i] = TRUE; precede[i] = nd; forward[i] = TRUE;
        x = c[nd][i] - f[nd][i];
        improve[i] = (improve[nd] < x) ? improve[nd] : x;
    } /*  fim de existir um nó i ... */
```

```
while (existir um nó i tal que (onpath[i] = FALSE) && (adjacent(i, nd) = TRUE)
        && (f[i][nd] > 0)) {
    /* o fluxo de i p/ nd pode ser diminuido. Coloque i no semicaminho */
    onpath[i] = TRUE; endpath[i] = TRUE; precede[i] = nd;
    forward[i] = FALSE;
    x = c[nd][i] - f[nd][i];
    improve[i] = (improve[nd] < f[i, nd]) ? improve[nd] : f[i][nd];
  } /*  fim de existir um nó i ... */
} /*fim de onpath[T]==FLASE */
if (onpath[T] == TRUE)
    achamos um semicaminho de S p/ T;
else
    o fluxo já é o ótimo
```

Assim que um semicaminho de S p/ T for encontrado, o fluxo poderá ser aumentado ao longo desse semicaminho (linha 6 da rotina principal) pelo seguinte algoritmo

```
x = improve[T];
nd = T;
while (nd != S) {
    pred = precede [nd];
    (forward[nd] == TRUE) ? (f[pred, nd] += x) : (f[nd, pred] -= x);
    nd = pred;
}
```