Ford-Fulkerson vs. Edmonds-Karp

Comparativo

Departamento de Ciência da Computação
Centro de Ciências Tecnológicas
Universidade do Estado de Santa Catarina

Luciani Regina da Costa Dantas
Priscila Sacae de Oliveira

Fluxo Máximo

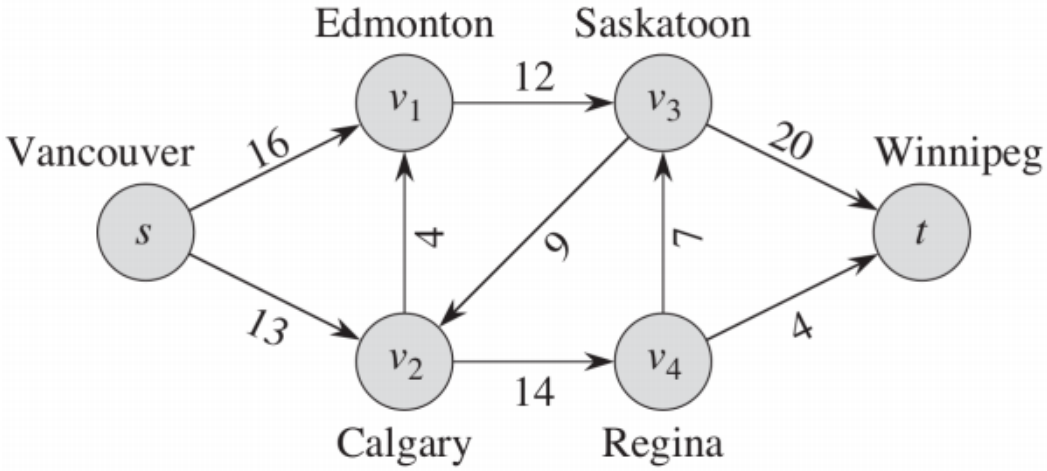
Grafos podem ser concebidos como uma rede de canalizações, onde o peso de cada aresta é a capacidade de cada canalização. O fluxo máximo de uma grafo é o maior fluxo que é enviado de um vértice inicial *s* até o vértice final *t*.

Restrições:

- f (u, v) fluxo na aresta (u, v);
- c(u, v) capacidade na aresta (u, v);
- Capacidades: $0 \leq f(u, v) \leq c(u, v)$ para qualquer (u, v);

Ford-Fulkerson

O algoritmo de Ford-Fulkerson consiste em achar por um **caminho aumentante** *p* (caminho pelo qual ainda é possível enviar fluxo) e aumentar um fluxo *f* de cada aresta do caminho *p* levando em conta o **grafo residual** (grafo que indica como podemos modificar o fluxo nas arestas de G depois de já aplicado o fluxo f).

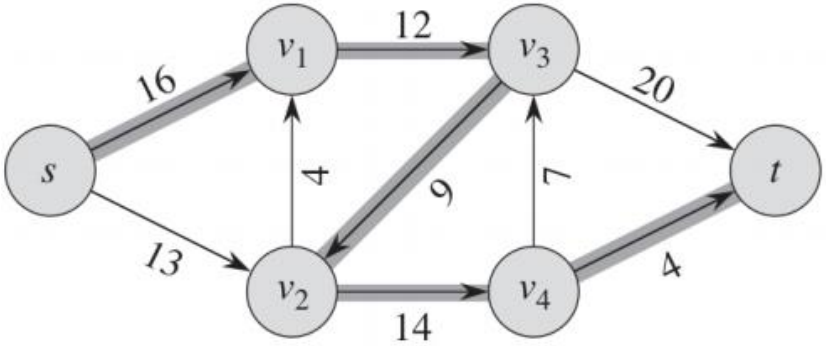


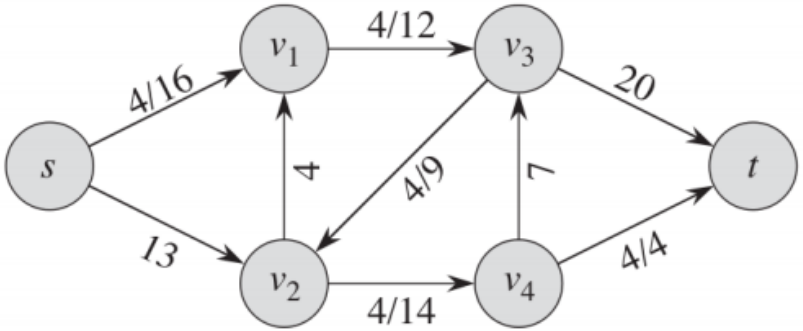
(imagem de Introduction to Algorithms, 3rd Edition)

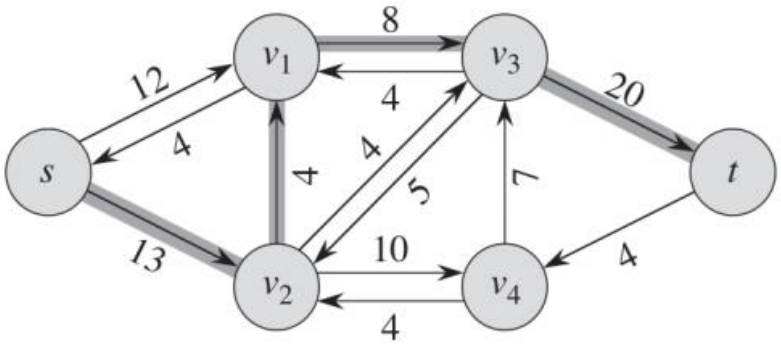
Ford-Fulkerson Pseudocódigo

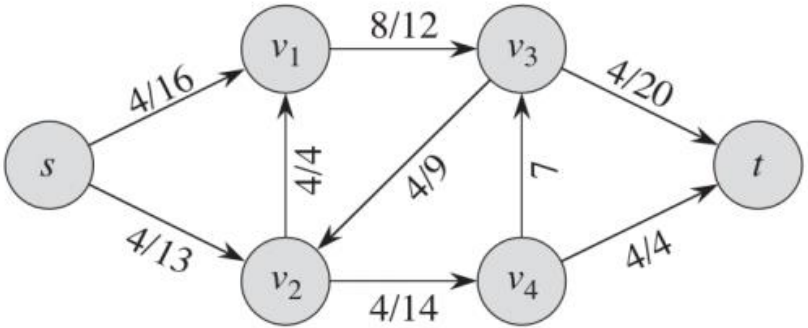
```
FORD-FULKERSON(G,s,t)
    para cada aresta (u,v) ← E[G]
        faça f[u,v] ← 0
             f[v,u] ← 0
    enquanto existir um caminho p de s até t na rede residual G_f
        faça c_f ← min{c_f(u,v):(u,v) está em p}
            para cada aresta em (u,v) em p
                faça f[u,v] ← f[u,v]+ c_f(p)
                     f[v,u] ← -f[u,v]
```

Ford-Fulkerson Passo-a-passo

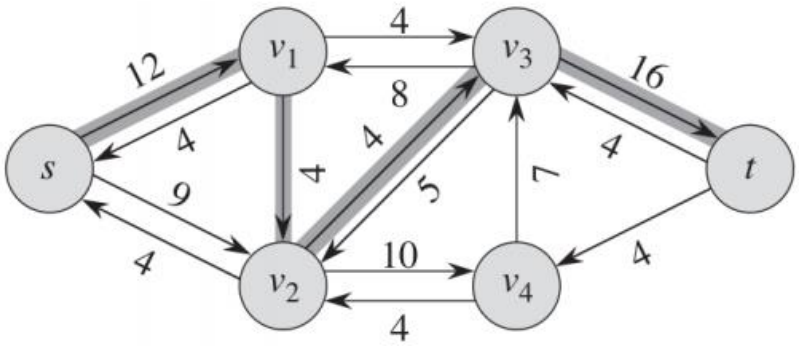


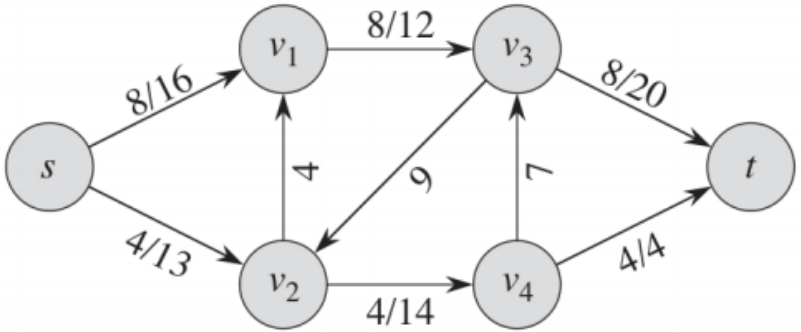


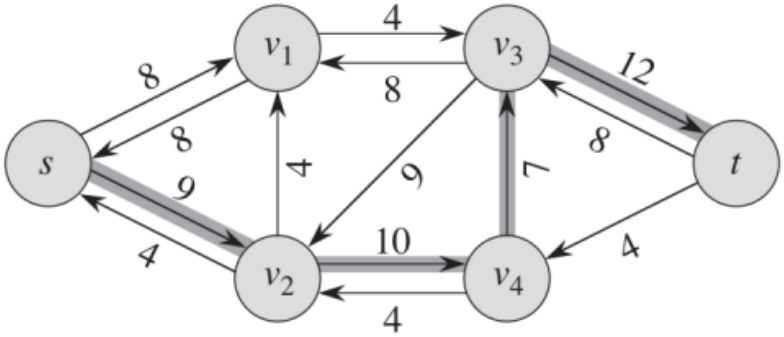


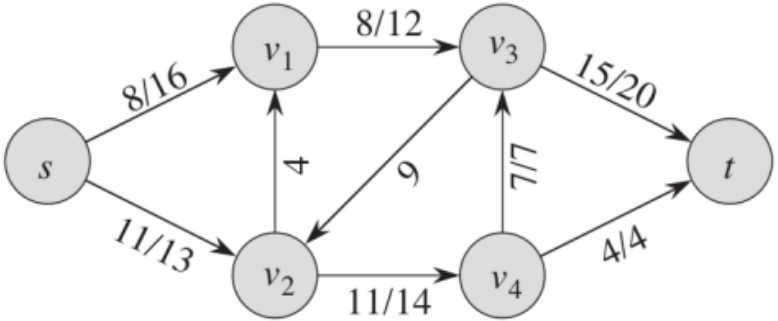


Ford-Fulkerson Passo-a-passo (cont.)

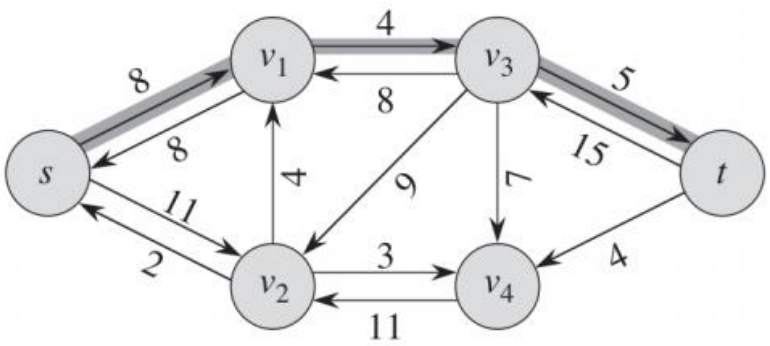


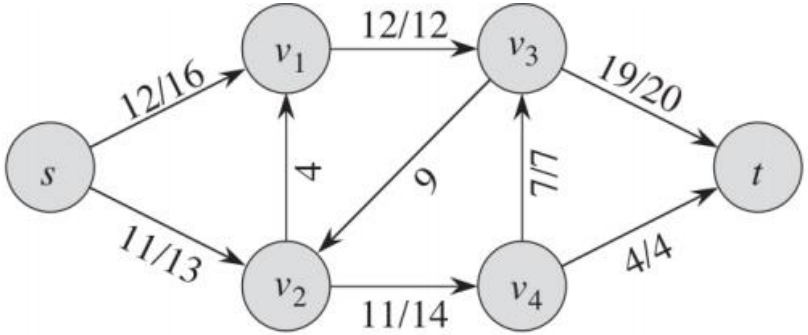


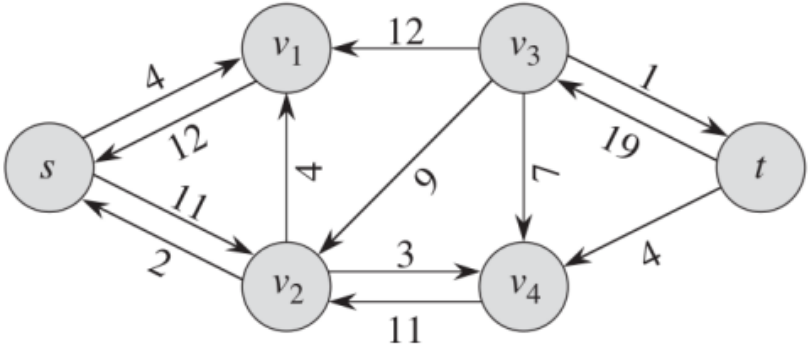




Ford-Fulkerson Passo-a-passo (cont.)

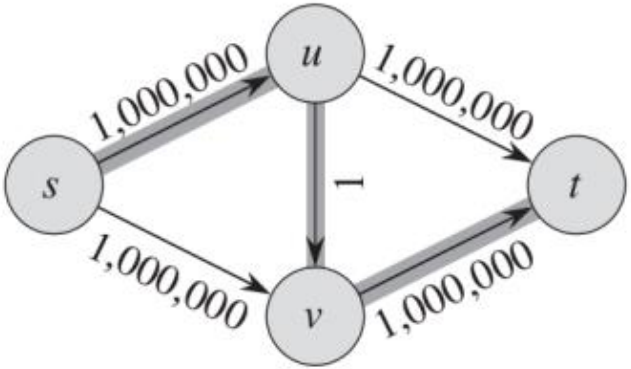


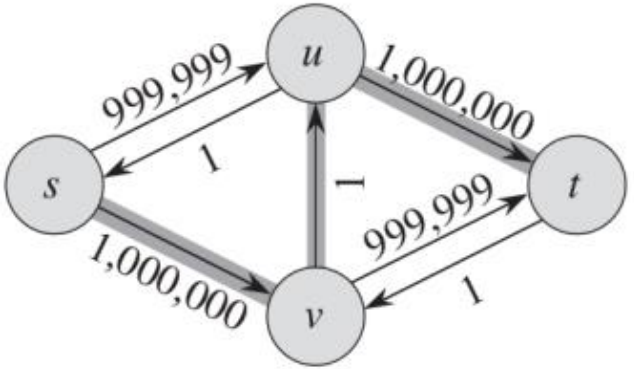




Ford-Fulkerson Complexidade de Tempo

A complexidade depende da resposta do fluxo máximo do grafo. No pior caso, será realizada uma iteração para cada unidade de fluxo do grafo, onde em cada interação é feita uma busca em todo o grafo. Como é realizada uma busca em profundidade, sua complexidade é **O(V+E)**, onde **V** é o número de vértice e **E** o número de arestas**.** Tendo **F** como o fluxo máximo do grafo, temos que **O((V+E)*F)**, admitindo que o grafo é conexo, temos que E ≥ V -1, podendo se resumir há **O(E*F).**





Ford-Fulkerson

Complexidade de Espaço

Para a busca dos caminhos é utilizada a busca em profundidade, sendo assim precisamos manter o vetor de nós visitados, logo a complexidade de espaço é **Θ(V)**, onde **V** é o número de vértices.

Edmonds-Karp

O algoritmo de Edmonds-Karp é uma implementação do algoritmo Ford-Fulkerson. A diferença está no fato do algoritmo procurar o caminho mais curto disponível entre s e t. Para encontrar esse caminho ele utiliza a pesquisa em largura. Dessa forma o algoritmo executa menos interações.

Edmonds-Karp Exemplo

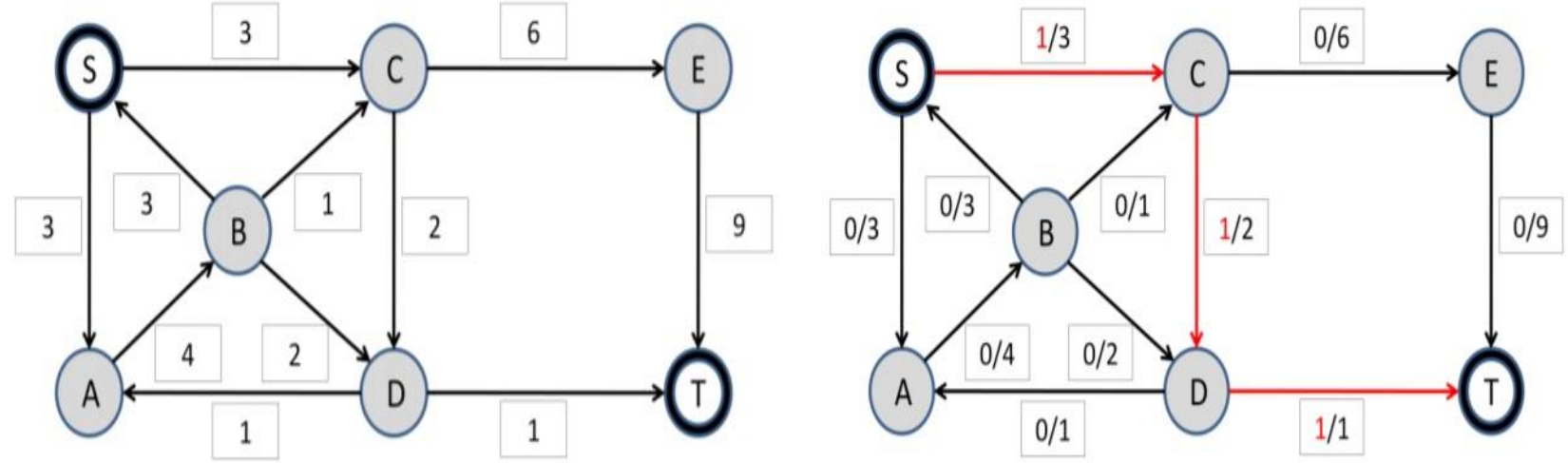
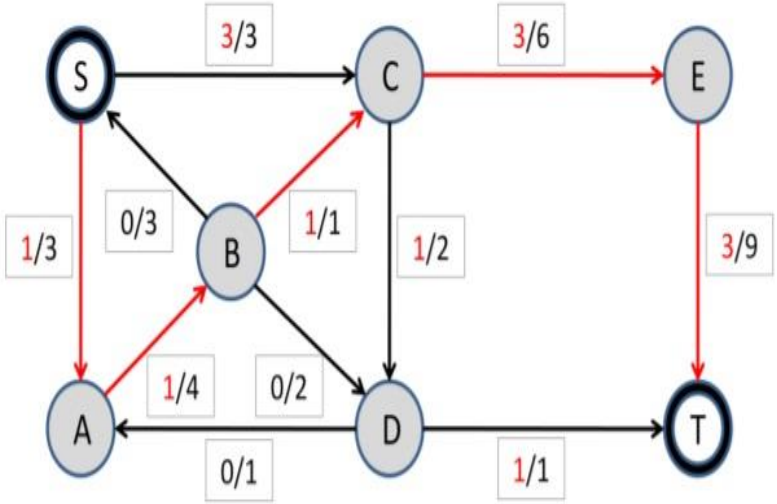


Figura: https://ppgcc.ufersa.edu.br/wp-content/uploads/sites/42/2014/09/victor-hugo-regis-de-freitas.pdf

Edmonds-Karp Exemplo



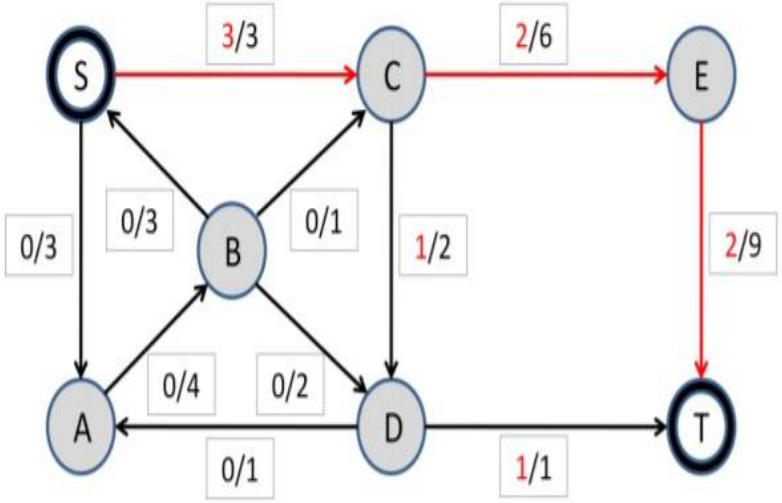
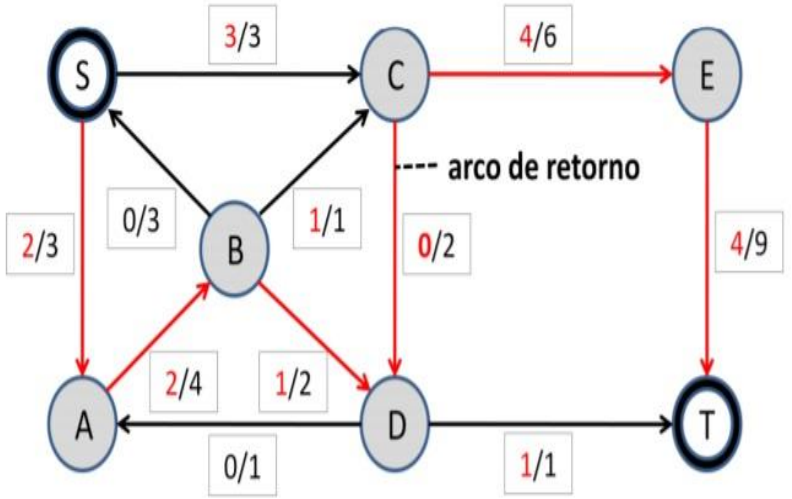


Figura: https://ppgcc.ufersa.edu.br/wp-content/uploads/sites/42/2014/09/victor-hugo-regis-de-freitas.pdf

Edmonds-Karp Exemplo



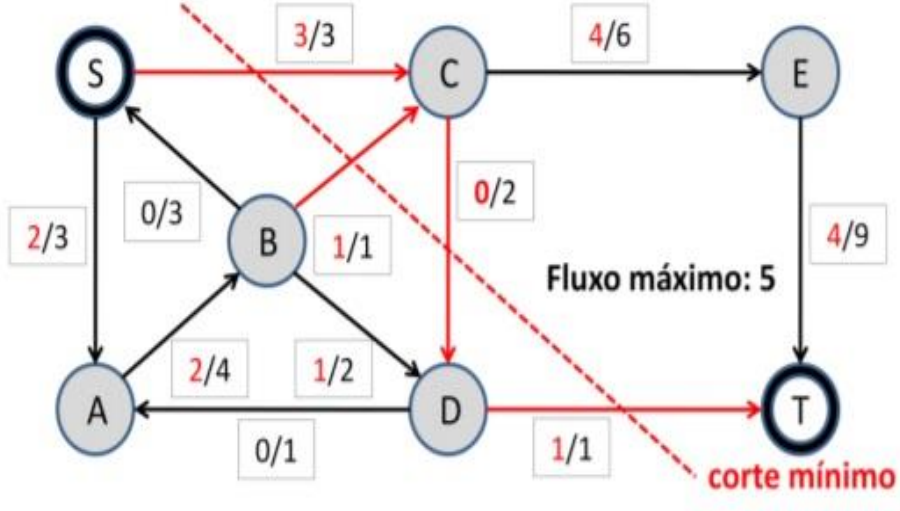


Figura: https://ppgcc.ufersa.edu.br/wp-content/uploads/sites/42/2014/09/victor-hugo-regis-de-freitas.pdf

Edmonds-Karp

Complexidade de Tempo

Edmonds- Karp remove a dependência do fluxo máximo de complexidade, tornando-o muito melhor para gráficos que tenham um fluxo máximo grande.

Executa cada interação em O(E) tempo e há no máximo V.E interações. A complexidade de tempo do algoritmo é $O(VE^2)$

Edmonds-Karp

Complexidade de Espaço

A busca dos caminhos é feita em largura, também precisa manter um vetor de nós visitados, logo a complexidade de espaço também é **Θ(V)**, onde **V** é o número de vértices .

Comparativo

Podemos verificar através das complexidades do algoritmos apresentados que o algoritmo de Edmonds-Karp é o mais eficiente:

- Não admite se as arestas forem números irracionais. (o algoritmo pode nunca terminar, demorar para encontrar fluxo máximo ou encontrar valor incorreto como resultado)
- O tempo de execução pode ser melhorado escolhendo em cada iteração o menor caminho aumentante entre *s* e *t*.