MAC121 - Algoritmos e Estruturas de Dados I

Universidade de São Paulo

Segundo Semestre de 2020

Fila de prioridade

Fila de prioridade

Problema: Considere, por exemplo, uma fila de processos que devem ser executados em um sistema operacional. Em cada passo desejamos executar o processo com menor (maior) prioridade da fila.

A estrutura de dados deverá realizar as seguintes operações eficientemente:

- Devolve o elemento de menor (maior) prioridade, e remove do conjunto;
- Insere um novo elemento com certa prioridade;
- Muda a prioridade de um elemento.

Fila de prioridade

Podemos implementar uma fila de prioridade utilizando um heap.

Cada posição do heap terá o elemento, e a prioridade dele.

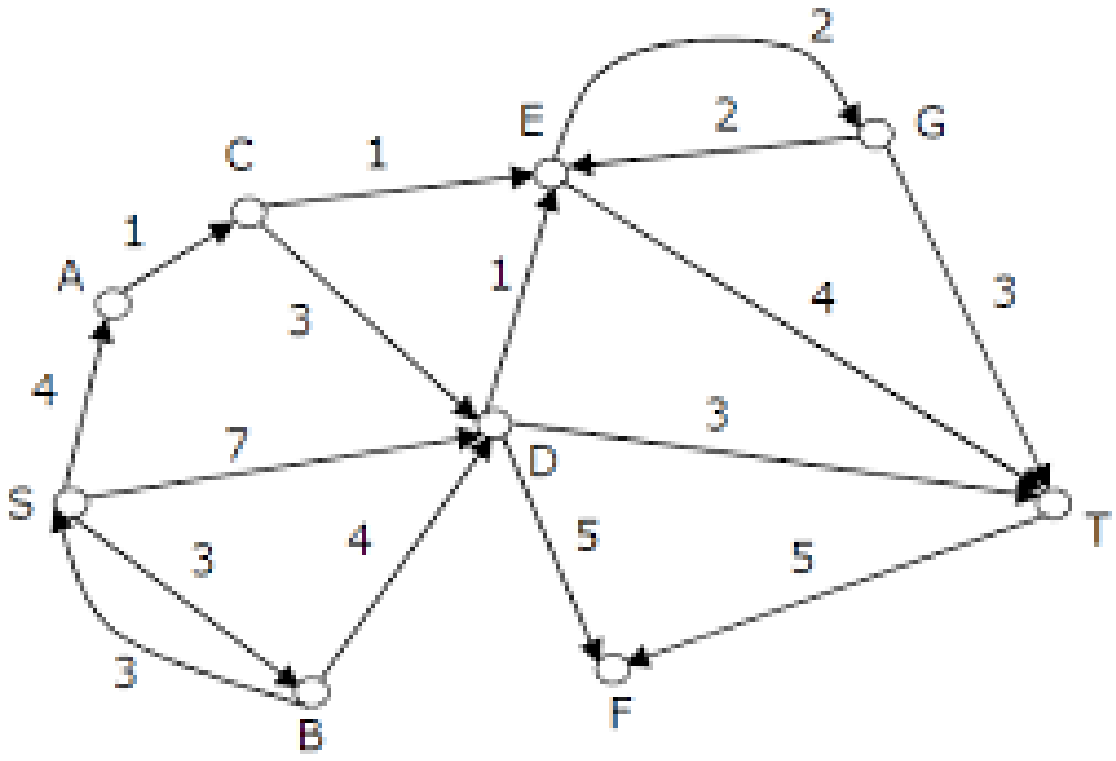
Além disso, para implementar eficientemente a operação de mudar a prioridade, precisamos guardar para cada item, em que posição do heap ele se encontra.

Fila de prioridade

```
typedef struct {
  int elem;
  int prior;
} item;

typedef struct {
  item *vet;
  int *ind;
  int n; /* elementos na fila */
  int max; /* tamanho da fila */
  int tam; /* total de elementos */
} filaPrior;

int removeMinimo (filaPrior * f);

void insere (filaPrior *f, item x);

void mudaPrior (filaPrior *f, int x, int novap);
```

Caminho mais curto

Aplicação de fila de prioridade: Caminhos mais curtos



Problema: Dado um conjunto de cidades, e estradas ligando estas cidades, e uma cidade *s*, determinar a distância de *s* a todas as outras cidades, usando o caminho mais curto.

Algoritmo de Dijkstra

Para resolver o problema usaremos um algoritmo parecido com o que apresentamos para o “problema do ratinho no labirinto”. Porém, ao invés de usarmos uma fila para guardar as posições atingidas, vamos usar uma fila de prioridade .

Em cada passo selecionamos a cidade que ainda não escolhemos que está mais próxima de s, pois para esta cidade já descobrimos a distância.

Este algoritmo foi inventado por E.W. Dijkstra em 1956, e funciona para o caso em que as distâncias entre as cidades são positivas.

Caminho mais curto com pesos positivos

```
lê grafo com n vért e m arestas com pesos
inicializa dist com ∞ para todos os vért
dist[s] = 0
insere s na fila de prioridade
enquanto a fila não está vazia
  Remove u o mínimo da fila de prioridade
  para cada vizinho v de u
    se dist[v] = ∞
      dist[v] = dist[u] + peso(u,v)
      insere v na fila
    senão se dist[v] > dist[u] + peso(u,v)
      dist[v] = dist[u] + peso(u,v)
      muda prioridade de v na fila
dist[v] tem o custo do caminho mínimo de s a v
```

Algoritmo de Dijkstra

O algoritmo de Dijkstra encontra o custo do caminho mínimo de s a todos os vértices de um grafo com n vértices e m arestas em tempo $O(m \log n)$.

Como encontrar os caminhos?