Algoritmos avançados – Exemplo de um plano de teste

1 Complexidade das operações

Objetivo: Verificar a complexidade das operações experimentalmente.
Vamos medir o tempo e o número de operações elementares. Uma operação elementar no heap binário é um swap, i.e. uma troca de chaves no sift-up ou sift-down.

Inserções (operação "insert") Fixa, por exemplo, $n = 2^{30} - 1$. Insere chaves $n, n-1, \ldots, 1$. Grava o tempo TA_i e o número SA_i de swaps depois de inserir $2^i - 1$, $i = 0, \ldots, 30$ chaves.

Para $i = 1, \ldots, 30$ o i-ésimo lote de inserções consiste da inserções $2^{i-1}, \ldots, 2^i - 1$. O número esperado de swaps para o lote é $SE_i = 2^{i-1}(i-1)$ e o tempo esperado $TE_i = O(SE_i)$. O número real de swaps para o lote é $S_i = SA_i - SA_{i-1}$ e o tempo real $T_i = TA_i - TA_{i-1}$.

Compara as quantidades empíricas e esperadas: S_i com SE_i e T_i com TE_i (por exemplo por plotar S_i/SE_i e T_i/TE_i) em função de 2^{i-1} e conclui.

Atualizações (operação "update") Para $i \geq 1$, insere $2^i - 1$ chaves de valor $2^i + 1$, seguido por 2^i chaves de valor $2^i + 2$. Depois atualiza as 2^i chaves de $2^i + 2$ para $2^i, 2^i - 1, \ldots 1$. Mede o tempo T_i e o número de swaps S_i das atualizações. Repete para $i = 1, \ldots, 20$.

O tempo esperado e o número esperado de swaps é $E_i = 2^i\, i$.

Compara S_i com E_i, plota T_i/E_i como função de 2^i, e conclui.

Deleções (operação "deletemin") Para $i \geq 1$, insere $n = 2^i - 1$ chaves aleatórias. Grava o tempo T_i e o número de swaps D_i para remover 2^{i-1} chaves. Repete para $i = 1, \ldots, 20$.

O tempo (pessimista) esperado e o número esperado de swaps é $E_i = 2^{i-1}(i-1)$.

Compara D_i com E_i, plota T_i/E_i como função de 2^i, e conclui.

2 Complexidade do algoritmo de Dijkstra

Objetivo: Comparar a complexidade teórica pessimista com a complexidade real. Em particular verificar que a complexidade real respeita o limite teórico.

Variação do número de arestas Fixa $n = 2^{20}$. Para $m = 2^i$, $i = 0, \ldots, 19$ roda o algoritmo de Dijkstra 30 vezes com um vértice inicial aleatória (sem parar no vértice destino) e grava n, m, o número de operações "insert" I, "deletemin" D e "update" U e o tempo num arquivo A.

Para avaliar verifica se $I \leq n$, $D \leq n$, $U \leq m$, e plota $T/(n+m)\log(n)$ em função de m e conclui.

Variação do número de arestas Fixa $m = 2^{20}$. Para $n = 2^i$, $i = 11, \ldots, 20$ roda o algoritmo de Dijkstra 30 vezes (nas mesmas condições do item anterior) e grava a mesmas quantidades num arquivo B.

Para avaliar verifica se $I \leq n$, $D \leq n$, $U \leq m$, plota $T/(n+m)\log(n)$ em função de n e conclui.

Aplica uma regressão linear $T(n,m) \sim an^b m^c$ nos dados completos dos arquivos A e B.

3 Comportamento em instâncias grandes

Objetivo: Avaliar o escalonamento do algoritmo Dijkstra.
Roda o algoritmo de Dijkstra 30 vezes com um vértice inicial aleatória nos grafos NY e EUA.
Relata o tempo médio e o consumo médio de memoria e conclui.