Kopce

Własności kopca

1. Pełne drzewo binarne
2. $A_{parent(i)} \geqslant A_i$
3. $h = \Theta(lg\, n)$

Podstawowe funkcje do obsługi kopca

```
parent( i )
  zwróć i/2

left( i )
  zwróć 2i

right( i )
  zwróć 2i + 1
```

przywracanie własności kopca

```
heap-down( i )                            // znany też jako down-heap albo heapify
  temp ← A[i]
  dopóki i ma lewego syna
    largest ← left(i)
    jeśli i ma prawego syna i A[right(i)] > A[left(i)]
      largest ← right(i)
    jeśli A[largest] > temp
      A[i] ← A[largest]
      i ← largest
    w przeciwnym przypadku
      zakończ pętlę
  A[i] ← temp
build-heap
  heap-size ← lenght
  dla i od lenght/2 do 1
    heapify( i )
```

kolejki priorytetowe

1. insert(x) – wstawia element x
2. maximum – zwraca element o największym kluczu
3. extract-max – usuwa maksymalny element i zwraca jego wartość

kopiec z dowiązaniami

problem: jak odwołać się do obiektu o danym indeksie, jeśli nie znamy jego pozycji w kolejce priotytetowej?

rozwiązanie: w kolejce priorytetowej trzymamy tylko numer elementu (indeks) i klucz, po którym sortuje kolejka; resztę informacji, włącznie ze wskaźnikiem do miejsca elementu w kolejce, trzymamy w zwykłej tablicy

Algorytm Dijkstry

idea algorytmu

Podobna do BFS, tylko przy kolejności odwiedzania wierzchołków uwzględniamy różne wagi krawędzi – dlatego wierzchołki trzymamy w kolejce priorytetowej zamiast zwykłej listy (kolejki).

implementacja

```
inicjalizuj
S ← ø
Q ← V
dopóki Q niepuste
  u ← Q.extract-min
  S ← S + {u}
  dla każdego wierzchołka v, takiego że (u,v) jest w grafie
    relaksuj krawędź (u,v)               // pamiętać o
przywracaniu własności kopca po relaksacji!
```

dowód poprawności

Niewprost. Niech u będzie pierwsze, dla którego $d[u] \neq \delta(s,u)$, gdy u jest wstawiane do S. Wiemy, że $u \neq s$, więc S jest niepusty przed wstawieniem u. W grafie musi istnieć ścieżka z s do u, bo inaczej $d[u]=\delta(s,u)=\infty$. Ponieważ s należy do S, a u nie należy do S, to $s \overset{S}{\rightarrow} x \rightarrow y \overset{V-S}{\rightarrow} u$.

Lemat: Gdy u jest dodawany do S, to $d[y]=\delta(s,y)$.

Dowód: $d[x]=\delta(s,x)$. W tym czasie nastąpiła relaksacja krawędzi (x,y). Własność relaksacji z poprzedniego dnia kończy dowód.

Ostatecznie, mamy $d[y]=\delta(s,y) \leqslant \delta(s,u) \leqslant d[u]$. Jednak skoro zarówno y, jak i u należą do V-S, to $d[u] \leqslant d[y]$. Dochodzimy do sprzeczności, bo

$d[y]=\delta(s,y)=\delta(s,u)=d[u]$.

złożoność

jeśli na tablicy - $O(V^2)$

jeśli na kopcu - $O((V+E)lgV)$ (dla grafów gęstych zwykła tablica jest wydajniejsza, natomiast dla grafów gęstych lepiej użyć kopca)

jeśli na kopcu Fibonacciego - $O(VlgV+E)$