





IFT2015 :: autumn 2017

Week 6: priority file - references and exercises

- the abstract type priority queue (*priority queue*)
- binary heap, heap d-ary, insertion and removal by *swim* and *sink*, heapisation
- sort by heap (*heapsort*)

References

(in non-exclusive disjunction)

-  [Sedgewick & Wayne 2011] §2.4 (<https://algs4.cs.princeton.edu/24pq/>)
-  [Sedgewick 2003] §9.1-§9.6
-  [Cormen, Leiserson, Rivest & Stein 2009] Chapter 6
-  wikipedia: *priority queue* (https://en.wikipedia.org/wiki/Priority_queue), *total order* (https://en.wikipedia.org/wiki/Total_order), *heap* / *heap* ([https://fr.wikipedia.org/wiki/Tas_\(informatique\)](https://fr.wikipedia.org/wiki/Tas_(informatique))), *d-ary heap* (https://en.wikipedia.org/wiki/D-ary_heap), *sort by heap* (https://fr.wikipedia.org/wiki/Tri_par_tas)
- Youtube: Robert Sedgewick Princeton / Coursera 1 (*priority queue*) (<https://www.youtube.com/watch?v=G9TMe0KC0w0>), 2 (*binary heap*) (<https://www.youtube.com/watch?v=YXCeM7w1630>), 3 (*heapsort*) (https://www.youtube.com/watch?v=H7rI5qtsf_Y); Srinivas Devas / MIT (*heaps and heapsort*) (<https://www.youtube.com/watch?v=B7hVxCmfPtM>)

1. Exercises

1.1 Binary heap with sentinels

We want to maintain a binary heap at indices $1 \dots n$ of a table $H[0 \dots L-1]$. Unused boxes on the left and right ($L > n + 1$) of the elements of the pile are used with sentinels: $H[0] = -\infty$, $H[n+1] = H[n+2] = \dots = H[L-1] = +\infty$. Here, $-\infty$ has a lower priority than any element and $+\infty$ has a higher priority than any element. Show a complete implementation with operations `deleteMin` and `insert` exploiting sentinels. The solution must also include the expansion / reduction of the dynamics underlying table: avoid overflow but ensure $L \leq c \times n$ with a reasonable constant (eg $c = 4$).

Index. The stopping conditions in the loops are simpler with sentinels: it is not necessary to test whether the index of the parent > 0 in `swim`, or if the index of the second child is always $\leq n$ in `sink`. However, the sentinels must be placed during initialization, deletion and reallocation.

1.2 Toddlers

Give an algorithm to enumerate elements with a priority less than or equal to an argument k in an array ordered as min-binary heap. The algorithm must take $O(m)$ time if the heap contains m such elements.

Index. Consider the binary tree representing the heap, and work out how to scroll through the recursion-requested items.

1.3 Binary priority file

Implement priority queue when there are just two possible values for the priority: 0 or 1. Any operation must execute in $O(1)$.

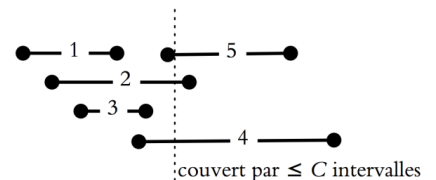
1.4 Minimax

On a un tableau $A[0..n-1]$ et un paramètre $0 < d \leq n$. On veut calculer $Q(A, d) = \max_i \min_{0 \leq j < d} A[i+j]$. Donner un algorithme qui calcule $Q(A, d)$ en temps $O(n \log d)$, avec $O(d)$ espace de travail au plus.

Indice. L'algorithme parcourt A en «glissant» une fenêtre de taille d au long du tableau: $m \leftarrow -\infty$; for $i \leftarrow 0, \dots, n-d$: $m \leftarrow \max\{m, M(i, d)\}$ où $M(i, d) = \min\{A[i], A[i+1], A[i+2], \dots, A[i+d-1]\}$ est la valeur minimale dans la fenêtre. Une solution naïve calcule $M(i, d)$ dans une boucle interne qui prend $O(d)$ temps. Une meilleure solution utilise une structure de données qui permet la mise à jour efficace $M(i, d) \rightarrow M(i+1, d)$, en temps $O(\log d)$.

1.5 Tri d'intervalles

On a une séquence d'intervalles $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ (stockée dans un tableau $T[0..n-1]$) avec $x_i < y_i$, triées par leur côté gauche: $x_1 < x_2 < x_3 < \dots < x_n$. Les intervalles peuvent être de longueurs différentes. On sait que partout il y a tout au plus C intervalles chevauchantes. Dans d'autres mots, pour tout z , il existe tout au plus C intervalles (x_i, y_i) avec $x_i \leq z < y_i$. Donner un algorithme pour trier les intervalles par leur côté droit. L'algorithme doit trier $T[]$ en temps $O(n \log C)$, avec un espace de travail de $O(C)$ au plus.




Remarque. Tri par insertion prendrait $O(n C)$ temps et $O(1)$ espace de travail.

1.6 Difficile à comparer

□ l'implantation usuelle du tas binaire, une insertion nécessite $O(\log n)$ comparaisons entre priorités et $O(\log n)$ affectations (de cases dans le tas). Montrer comment faire l'insertion avec $O(\log \log n)$ comparaisons et $O(\log n)$ affectations. (Une telle solution est utile quand la comparaison est plus coûteuse que l'affectation —p.e., avec des chaînes de caractères.)

Indice. Considérez les indices visités dans la boucle de swim entre la dernière case et la racine. Identifiez rapidement où la boucle va arrêter.

Advertisements



ARMANI EXCHANGE BLA...

the slim fit casual jean is garment dyed with a classic five pocket denim style. comfortable fit with...

\$45

Buy Now

ARMANI EXCHANGE KHA...

ARMANI EXCHANGE BLA...

the slim fit casual jean is garment dyed with a classic five pocket denim style. comfortable fit with...

\$45

Buy Now

ARMANI EXCHANGE BLA...

□ novembre 8, 2017 □ csurosm □ exercice, référence

Une réflexion sur “Semaine 6: file de priorité – références et exercices”

1. SGV dit :

novembre 9, 2017 à 7:29

Est-ce possible de savoir la moyenne pour l'examen intra? Merci d'avance

Répondre

