

## Быстро растет бамбук...–2

### Решение .

Решение задачи, по крайней мере в нулевом приближении, просто и ясно: срезанный бамбук надо продавать в тот день, когда цена на него максимальна. Поскольку через  $K$  дней получить деньги за бамбук невозможно, то максимальная цена на бамбук, срезанный в  $i$ -й день, – это просто максимальная из цен на бамбук в  $i$ -й,  $(i+1)$ -й,  $(i+2)$ -й, ...,  $(i+(K-1))$ -й дни. Итого, нам надо вычислить

$$\sum_{i=1}^N L_i \cdot \max_{i \leq j \leq i+K-1} P_j,$$

где  $L_i$  – количество срезанного в  $i$ -й день бамбука, а  $P_j$  – стоимость бамбука в  $j$ -й день.

Конечно, надо учесть, что сезон роста бамбука заканчивается через  $N$  дней, т.е. при  $j > N$  величина  $P_j$  не определена (если эти величины нам понадобятся, то можем считать  $P_j = 0$  при  $j > N$ ), но это уже мелочи. Гораздо неприятнее то, что если реализовывать эту формулу прямо в лоб, то нам понадобится  $N$  раз вычислять максимум  $K$  чисел, т.е. мы получим сложность  $O(N \cdot K)$ , а это неприемлемо много для данных ограничений.

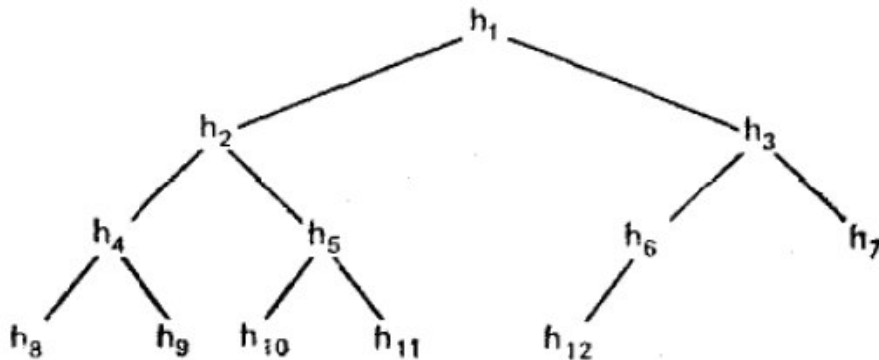
Итак, мы столкнулись в этой задаче с очень распространённой ситуацией: имеется множество, которое может изменяться с течением времени – некоторые элементы множества уходят из него, зато могут появляться новые элементы. Каждый элемент имеет некоторую характеристику. Нам нужно в некоторые моменты жизни этого множества определять элемент с наибольшей (наименьшей, наилучшей, наизамечательнейшей и т.д.) характеристикой.

Для разрешения подобных ситуаций хорошо подходят различные *пирамидальные* структуры (или *кучи*). В данном случае применим *бинарную кучу*.

В бинарной куче элементы образуют бинарное дерево, в котором

1. каждый уровень, кроме, может быть, последнего, заполнен – т.е. содержит максимальное возможное количество узлов;
2. последний уровень заполняется слева направо, пока не закончатся элементы.

На рисунке изображена бинарная куча из 12 элементов:



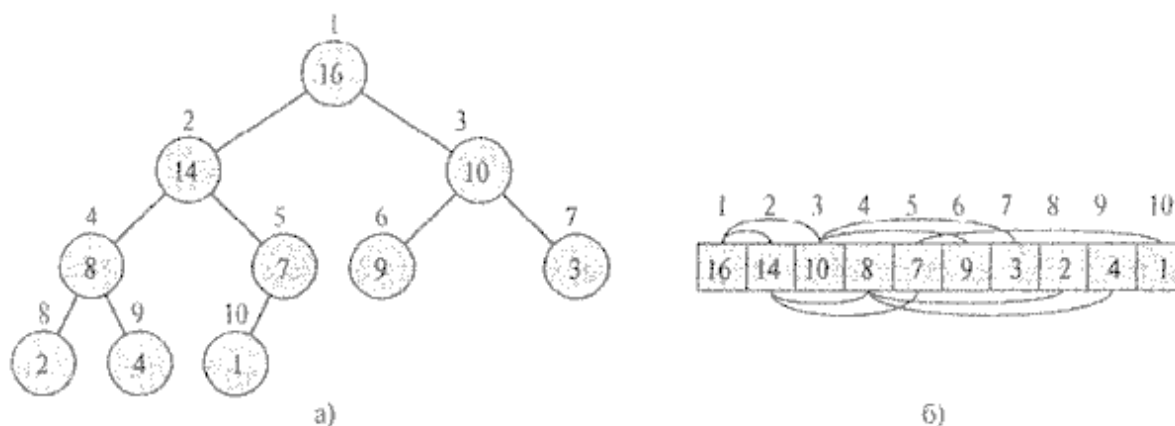
Представлять бинарную кучу удобно с помощью массива, располагая элементы в соответствии с индексами на рисунке. Легко видеть, что предок

элемента  $h_j$  располагается в  $h_{\lfloor j/2 \rfloor}$  (как обычно запись  $\lfloor x \rfloor$  обозначает наибольшее целое число, не превосходящее  $x$ ), его левый потомок – в  $h_{2j}$ , правый – в  $h_{2j+1}$ .

Кроме того, для наведения порядка ☺, потребуем чтобы куча была неубывающей: характеристика каждого элемента кучи должна быть больше или равна характеристикам его потомков.

Приведу пример, заимствованный из не раз уже упоминавшейся книги

*Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-е изд. 1296 стр. М.: Вильямс, 2006:*



Пирамида, представленная в виде а) бинарного дерева и б) массива

Соответствующий фрагмент этой книги выложен в учебных материалах. Вообще, по бинарным кучам имеется громадное количество материала. В учебных материалах я выкладываю фрагменты из четырёх классических книг:

1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов, 536 стр. М.: Мир, 1979.
2. Вирт Н. Алгоритмы и структуры данных, 360 стр. М.: Мир, 1989.
3. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-е изд. 1296 стр. М.: Вильямс, 2006.
4. Сэдзвик Р. Фундаментальные алгоритмы на C++. Часть 1-4, 687 стр. М., Диасофт, 2001

И тогда решение задачи выглядит примерно так:

1. Инициализация.

- 1.1. Создать бинарную кучу из величин  $P_1, P_2, \dots, P_K$ . Например, последовательно вставляя в неё  $P_1, P_2, \dots, P_K$ .
- 1.2. Установить результат равным 0.

2. Цикл. Для всех  $i$  от 1 до  $N$  сделать следующее:

- 2.1. Увеличить результат на  $L_i \cdot h_i$ . Заметим, что  $h_i$  – это максимальный элемент кучи.
- 2.2. Заменить в куче элемент, равный  $P_i$  на  $P_{i+K}$ . Если  $(i+K) > N$ , то считать  $P_{i+K} = 0$ .

Конечно, остаются вопросы: как добавить элемент в кучу, как заменить элемент в куче, как найти местоположение элемента  $P_i$  в куче.

Ответ на первые два вопроса стандартный – ниже приведены две процедуры, выполняющие соответствующие действия. Комментировать их я не буду – в приведённых учебных материалах всё рассматривается очень подробно.

```
ADD(x)  // добавить в N-элементную кучу число x
  N:= N+1;
  h[N]:= x;
  i:= N;
  While (i > 1) and (h[i] < h[i div 2]) Do Begin
    S:= h[i];
    h[i]:= h[i div 2];
    h[i div 2]:= S;
    i:= i div 2;
  End;
End ADD;

CHANGE(i, x) // изменить величину h[i] на x
  h[i]:= x;
  If (i > 1) and (h[i] < h[i div 2]) Then Begin
    // 1 случай: новое число поднимается
    While (i > 1) and (h[i] < h[i div 2]) Do Begin
      h[i]:= h[i div 2];
      h[i div 2]:= x;
      i:= i div 2;
    End;
  End
  Else Begin
    // 2 случай: новое число опускается
    Repeat
      If (i >= 2*N) Then lson:= h[2*i]
                      Else lson:= -∞;
      If (i >= 2*N+1) Then rson:= h[2*i+1]
                      Else rson:= -∞;
      If lson >= rson Then j:= 2*i
                      Else j:= 2*i + 1;
      If h[i] >= h[j] Then Break;
      h[i]:= h[j];
      h[j]:= x;
      i:= j;
    Until False;
  End
End CHANGE;
```

Легко видеть, что высота бинарного дерева с  $K$  вершинами есть  $O(\log K)$  – собственно это то, ради чего мы и заходились с бинарным деревом. Соответственно, легко видеть, что операции вставки и замены требуют  $O(\log K)$  времени.

Что же касается вопроса, как найти местоположение элемента  $P_i$  в куче, то тут ответ простой – завести массив, в котором для каждого  $i$  хранить местоположение элемента  $P_i$  в куче, и одновременно с перемещениями элементов по куче отображать эти перемещения в массиве местоположений. Конечно, в куче удобнее хранить не сами элементы  $P_i$ , а их индексы, а элементы  $P_i$  хранить в своём массиве. В общем, простая техника, требующая только аккуратности в реализации.

В результате получаем решение со сложностью  $O(N \cdot \log K)$  с затратами памяти  $O(N)$  – это то, что надо.

И последние замечания. Во-первых, вышеописанный способ создания кучи требует  $O(K \cdot \log K)$  времени. Однако кучу можно создать быстрее – за  $O(K)$ . В данной задаче это не очень существенно, а вообще может оказаться важным. Очень подробно этот процесс описан в пункте 6.3 многократно упоминаемой книги *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ*, 2-е изд. 1296 стр. М.: Вильямс, 2006 – смотрите учебные материалы. Во-вторых, легко видеть, что при реализации передвижений числа  $x$  по куче, совсем не обязательно всё время присваивать  $x$  элементу кучи: вполне достаточно делать это один раз – в конце операции. Мне показалось, что так изложение будет понятнее, а усовершенствование процедур с учётом этого замечания послужит прекрасным, хоть и небольшим, упражнением.