Solutions of Homework4

2011013251 Lv Wanqi

March 25, 2013

Solution of 15.5-4

As Knuth has proved, $root[i, j-1] \leq root[i, j] \leq root[i+1, j]$. Thus we can modify the *Optimal - BST(p, q, n)* algorithm on the 9th line ($r \leftarrow i\ to\ j$) and replace it with the following line:

$$r \leftarrow root[i, j-1]\ to\ root[i+1, j]$$

Therefore the time complexity of the modified algorithm is:

$$T(n) = \sum_{l=1}^{n} \sum_{i=1}^{n-l+1} (root[i+1, i+l-1] - root[i, i+l-2] + 1)$$

$$= \sum_{l=1}^{n} (root[n-l+2, n] - root[1, l-1] + n - l + 1)$$

$$\leq \sum_{l=1}^{n} (2n)$$

$$= O(n^2)$$

While we have $T(n) \geq 2 \times (n + (n-1) + \cdots + 1) = \Omega(n^2)$
Thus we have $T(n) = \Theta(n^2)$

Solution of 15-3

Idea:
First we sort the n points by x-coordinate, using an algorithm of $\Theta(nlgn)$, and label them from 1 to n.
For a bitonic tour, we can divide it into 2 paths, i.e. tours that start at the leftmost point(*point*1), go strictly rightward to the rightmost point(*point n*), and then go strictly leftward back to the starting point.
If we make the 2 paths both start at *point*1, then we have 2 increasing sequences. Assume that the first path ends at *point* i while the second path ends at *point* j and the shortest path is $m(i, j)$, and obviously $m(i, j) = m(j, i)$. Thus we have that $min(n, n)$ is the targeted minimum length.
i)
if $i = j$, there must be one end of the 2 paths which is linked to *point* $i - 1$, and the other linked to *point* w ($1 \leq w \leq i - 2$), so we have
$\mathbf{m(i, i) = min\{m(i-1, w) + d(i-1, i) + d(w, i)\}}$

ii)
if $j = i+1$, then *point* $i+1$ is linked to *point* w ($1 \le w \le i-1$), so we have $\mathbf{m(i,j) = min\{m(i,w) + d(w, i+1)\}}$
iii)
if $j > i+1$, then we have $\mathbf{m(i,j) = m(i, j-1) + d(j-1, j)}$
pseudocode:

Optimal-Bitonic-Tour

```
      ▷ the variable n is the number of the points
      ▷ c[i] is point i
      ▷ d(i, j) is the distance of p[i] and p[j]
  c = sort points by x
  m(0, 0) = 0;
  for i ← 1 to n
      for j ← 1 to n
         do
            if i = j
              then
                    for w ← 1 to i − 2
                        do
                           m(i, j) = minm(i − 1, w) + d(i − 1, i) + d(w, i)
                           c(i, j) = argmin(m(i − 1, w) + d(i − 1, i) + d(w, i))
      else  if j = i + 1
              then
                    for w ← 1 to i − 2
                        do
                           m(i, j) = minm(i, w) + d(w, i + 1)
                           c(i, j) = argmin(m(i, w) + d(w, i + 1))
      else  if j > i + 1
              then
                    m(i, j) = m(i, j − 1) + d(j − 1, j)
                    c(i, j) = j − 1
```

Analysis:
Given that no two points have the same x-coordinate and that all operations on real numbers take unit time, we have $T(n) = \Theta(n^2)$.

Solution of 16.2-6

Idea:
1. find the median **value[k]/weight[k]** among all **value[i]/weight[i]** in linear time, and then divide the set of goods **G** into three subset $\mathbf{G_1, G_2, G_3}$ and $\mathbf{G = G_1 \cup G_2 \cup G_3}$

$G_1 = \{g_i : value[i]/weight[i] > value[k]/weight[k]\}$

$G_2 = \{g_i : value[i]/weight[i] = value[k]/weight[k]\}$

$G_3 = \{g_i : value[i]/weight[i] < value[k]/weight[k]\}$

meanwhile, calculate the sum weight of each subset $\mathbf{W_1, W_2, W_3}$ in linear time and then :
i) if $W_1 > W$, then $G \leftarrow G_1$, do recursion;
ii) if $W_1 \le W$ && $W_1 + W_2 \ge W$, then put G_1 all in and randomly put goods with the weight of $W - G_1$ in G_2 in , exit;
iii) if $W_1 \le W$ && $W_1 + W_2 < W$, then put G_1, G_2 all in , $G \leftarrow G_3, W \leftarrow W - W_1 - W_2$, do recusion; **Pseudocode:**

FRACTIONAL - KNAPSACK

k = median(G)
partition(G, k, G_1, G_2, G_3)
W_1 = totalWeight(G_1)
W_2 = totalWeight(G_2)
W_3 = totalWeight(G_3)
if $W_1 > W$
  then
    $Fractional - knapsack(G_1, W, P)$
  else
if $W_1 \le W$&& $W_1 + W_2 \ge W$
  then
    $P \leftarrow P \cup G_1$
    $W = W - W_1$
    for each g_i in G_2
    if $W > w_i$
      then
        $P \leftarrow P \cup g_i$
        $W + W - w_i$
      else
        $P \leftarrow P$\{part of g_i with the weight of W\}
  else
    $P \leftarrow P \cup G_1 \cup G_2$
    $W = W - W_1 - W_2$
    Fractional -knapsack(G_2, W, P)

Analysis: $T(n) \le T(\frac{n}{2} + \Theta(n)$, thus $T(n) = \Theta(n)$