算法设计 HW04

Xun Ying

2024 年 7 月 8 日

Q1 (a) First, we assume f_i is the biggest revenue sum with (1,1)-step and a_i is the last element of f_i. So what we want to find is $\max f_i$

We already know $f_0 = 0$, and $i_{j+1} = i_j + 1$,so we can easily get that $f_i = max(f_{i-1}, 0) + a_i$. We scan from 1 to n and get the result.

The time complexity is $O(n)$, as we just scan once and every time just do a comparison.

To prove the correctness, we can use induction. First, we already know that $f_i = \max(a_1, 0)$ is correct. Then we assume f_{i-1} is correct. And we consider all subsequences whose last element is a_i, so, it either just a_i if $f_{i-1} < 0$, or $f_{i-1} + a_i$ if $f_{i-1} > 0$, which is just the equality above. So we can get that f_i is correct. By induction, the alogrithm is correct.

(b) Just like above, we set f_i, and we know that $f_0 = 0$.

And now, we we can find $f_i = \max_{i-R \le j \le i-L} \max(f_j, 0) + a_i$

The time complexity is $O(n^2)$, as every scan we must traverse from $i - L$ to $i - R$

The correctness is like above by induction.

(c) We can improve the alogrithm by monotonic queue. We maintain a queue Q . And for every j, we pop out index i which $i < j - R + L$. Then if $f_j > 0$, we push it into Q, the top of Q is the biggest f_j from $j - R + L$ to j. And we can get f_{j+L} by $O1$, so the total time complexity is $O(n)$.

Q2 Let $f_{i,j}$ be the minimum number of comparisons for $a_i, a_{i+1}...a_j$, so what we want is $f_{1,n}$. And we know that $f_{i,i} = \omega_i$.

Then, if $i < j$, we assume a_k is the tree root. So the total number of comparisons is $f_{i,j} = \min_{i \le k \le j}(f_{i,k-1} + f_{k+1,j}) + \sum_{t=i}^{j} \omega_t$

The time complexity is $O(n^3)$, as we must traverse all the possible k.

The correctness is like above by induction. First, if tree has only one node, the result is correct. Then we assume every $f_{i,j}$ which $j - i < d$ is correct, and we consider $f_{i',j'}$ which $j' - i' = d$ all the possible k, so that we can consider every possible tress. And for every possible tree, we divide it into three parts and every part has been solved already. If we merge two trees, the total number of comparisons will increase $\sum_{t=i'}^{j'} \omega_t$ as every tree node's depth will increase 1. So we can get that equality above is correct. So we can get that $f_{i',j'}$ is correct. By induction, the alogrithm is correct.

Q3 We set $f(S, u)$ to be whether T(u) can be colored by the color in S, which is the color set of vertices in $B(u)$.

Then consider every subtree$T(v_i)$ divided by $B(u)$ and $B(T(v_i) \backslash B(u))$ are independent. So, we can enumerate every color set $T \subset S$ and T contains the color in $B(v_i \cap B(u))$ and then we can determine whether there exists a T which make $f(S, v_i) = 1$. At last, we just need to check whether $f(S, v_0) = 1$.

The time complexity is $O(k^2 c^{2k} n)$ as we must enumerate every vertex and for each vertex we must enumerate every possible color set and change state.

The correctness is easy. First a lonely leaf is correct. Then just like above we use induction to prove.

Q4 (a) It's hard to prove the equality, so we just use $\le$.

First, consider the logest path. We choose s+1-th vertex u, so we have that $d^r(i, j) = d^s(i, u) + d^t(u, j) \le \max_{i \le k \le j} d^s(i, k) + d^t(k, j)$

Then, we can enumerate every k and construct a path from i to k and from k to j with $d^s(i, k) + d^t(k, j)$ The path is not longer than the longest path. So we can get that $d^s(i, k) + d^t(k, j) \le d^r(i, j)$

So we can get that $d^r(i, j) = \max_{i \le k \le j} d^s(i, k) + d^t(k, j)$

(b) It is a convex polygon, so $v_i, v_j, v_{i'}, v_{j'}$ also forms a convex polygon. So, $v_i v_j$ and $v_{i'} v_{j'}$ must intersect. We call it k

So we have $d(i,j) = d(i,k) + d(k,j)$, and $d(i',j') = d(i',k) + d(k,j')$ So we can get that $d(i,j) + d(i',j') = d(i,k) + d(k,j) + d(i',k) + d(k,j') \geq d(i,j') + d(i',j)$

(c)We can make an induction on r

First, r=1 has been proved by (b).

Then, assume $r = k$ is right. Considering $r = k+1$, assume the k-th vertex in $d^{k+1}(i',j)$ is u, and the k-th vertex in $d^{k+1}(i,j')$ is v. So we have $d^{k+1}(i',j) = d^k(i',u) + d(u,j)$ and $d^{k+1}(i,j') = d^k(i,v) + d(v,j')$

If $u \leq v$, we have $d^{k+1}(i,j) \geq d^k(i,u) + d(u,j)$ and $d^{k+1}(i',j') \geq d^k(i,v) + d(v,j')$ by (a). And by $r = k$ is right, we can have that

$$\begin{aligned}
&d^{k+1}(i,j) + d^{k+1}(i',j') \\
\geq &d^k(i,u) + d^k(i',v) + d(u,j) + d(v,j') \\
\geq &d^k(i',u) + d^k(i,v) + d(v,j) + d(u,j') \\
\geq &d^{k+1}(i,j') + d^{k+1}(i',j)
\end{aligned}$$

If $u > v$, just like above. As a result, $d^r(*,*)$ satisfies iQI too.

(d) For the first, we let $u = K^r(i,j), v = K^r(i,j+1)$.

Assume $u > v$, then$i \leq v < u \leq j \leq j+1$. So $d^t(v,j) + d^t(u,j+1) \geq d^t(u,j) + d^t(v,j+1)$ and $d^s(i,v) + d^t(v,j+1) \geq d^s(i,u) + d^t(u,j+1)$. So, $d^s(i,v) + d^t(v,j) \geq d^s(i,u) + d^t(u,j) = d^r(i,j)$,which is contradiction. As a result, $u \leq v$.

For the second, we let $w = K^r(i+1,j+1)$

Assume $v > w$, then $i < i+1 \leq w < v$. So $d^s(i,w) + d^s(i+1,v) \geq d^s(i+1,w) + d^t(i,v)$ and $d^s(i+1,w) + d^t(w,j+1) \geq d^s(i+1,v) + d^t(v,j+1)$ So, $d^s(i,w) + d^t(w,j+1) \geq d^s(i,v) + d^t(v,j+1) = d^r(i,j+1)$, which is contradiction. As a result, $v \leq w$.

Therefore, we can get that $K^r(i,j) \leq K^r(i,j+1) \leq K^r(i+1,j+1)$

(e) By (d), we can calculate d^r by enumerate$d^s and d^t$ in $O(n^2)$. So, the alogrithm is to calculate$d^1, d^2 ... d^{\log r}$, then we can get d^r by binary representation of r.

So the total time complexity is $O(n^2 \log r)$.

Q5 difficulty: Q1 and Q2 are 3, Q3 and Q4 are 5 Collaborator: 蒋松霖