Algorithm Homework 1

龙肖灵
Xiaoling Long
Student ID.:81943968
email:longxl@shanghaitech.edu.cn

October 8, 2017

Problem 1

Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

$$g_1(n) = 2^{\sqrt{\log n}}$$
$$g_2(n) = 2^n$$
$$g_4(n) = n^{4/3}$$
$$g_3(n) = n(\log n)^3$$
$$g_5(n) = n^{\log n}$$
$$g_6(n) = 2^{2^n}$$
$$g_7(n) = 2^{n^2}$$

Solution. After plotting all function in Matlab. I found the answer.

The order is $g_1(n) < g_4(n) < g_3(n) < g_5(n) < g_2(n) < g_7(n) < g_6(n)$.

■

Problem 2

The following program computes the $n-th$ power of integer x.

```
function exp(x, n)
  res = 1
  for i = 1 to n
    res *= x
return res
```

(a) For some function f that you should choose, give a bound of the form $O(f(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm).

(b) The algorithm above is quite simple. Can you do better? Design another algorithm and show that its running time is $O(g(n))$, where $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$

Solution. There are two question.

1. In the code above, there are just a cycle from 1 to n and two assignment operation. Hence $f(n) = 2n + 1$, the running time $O(f(n)) = O(n)$.

2. I design a iteration algorithm as following.

```
res = 1                                          1
function exp(x, n)
  if(n % 2 != 0)
    res *= x                                     1
    n = n - 1                                    1
  if(n = = 0)
    return res
  else
    res *= exp(x, n/2) * exp(x, n/2)             2
```

   We get that $g(n) = 4\log_2 n + 1$. Then

$$\begin{aligned}\lim_{n\to\infty}\frac{g(n)}{f(n)} &= \lim_{n\to\infty}\frac{4\log_2 n+1}{2n+1}\\ &= \lim_{n\to\infty}\frac{\frac{4}{n}}{2}\\ &= \lim_{n\to\infty}\frac{2}{n}\\ &= 0\end{aligned}$$

   Hence, the algorithm designed satisfies the requirement in the question.

Done. ■

Problem 3

During the National Day holiday, scenic areas are always full of people. Bravely, you are planning to visit such a scenic area in the coming holiday. In that place there are n famous scenic spots, all of which you want to visit. But in order to manage the traffic flow, roads between these spots can be only got through in one direction. Now you get the information of m roads and their traffic directions from the tourist map. Suppose there is not a cycle in this map. Design an $O(m + n)$ algorithm to find if there exists such a path that you can visit all the n spots exactly once and if so, output such a path. You can start at whichever spot.

Solution. This is a directed graph traversal. My algorithm as following

First step, count the number of the road pointing to of every scenic area(called pointing_number).

Second step, find the spot with the least pointing_number as starting spot.

Third step, find the spot with the least pointing_number from the child spot of last spot added to path. If the least pointing_numeber is 0, so there doesn't exist a path. If not do the step again till the path include all spots.

Finally, we can get the answer of if there exists such a path that I can visit all the n spots once.

Running time of the algorithm above $f(m, n) = 3n + 2m$. Hence $O(f(n)) = O(m + n)$. ■

Problem 4

Suppose a scenic area has n famous scenic spots with m undirected roads between them, and the n spots are connected. Because of the complaints from tourists during the holiday, now the officers want to add a public toilet in one of the n scenic spots. Such a spot will close during the construction period, which means tourists are forbidden to visit the spot. How can you find such a spot without which the tourists can still visit all the other $n-1$ spots starting from any of them (i.e., they are still connected). Give an $O(m+n)$ algorithm to achieve the goal.

Solution. This problem is associated with connectivity of graph.

First, run BFS on graph. Then we can choose which ones we can add a toilet in. So if degree of the root of BFS tree equals 1, the root spot can built toilet. If not we can built toilet in any one of the leaf node sof spot. And we know that BFS runs in $O(m+n)$ time if the graph is given by its adjacency list representation. So this algorithm satisfies the requirement of question. ■

Problem 5

Let $G = (V, E)$ be an n-node undirected graph containing two nodes s and t such that the distance between s and t is strictly greater than $n/2$. Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all $s-t$ paths. In other words, the graph obtained from G by deleting v contains no path from s to t. Give an algorithm with running time $O(m+n)$ to find such a node v.

Solution. Let's prove the existence of node v first.

It must be a cycle contain node s and t, if node v connot destroy all $s-t$ path. And the distance of $s-t$ is strictly greater than $n/2$. The distance greater than $n/2$ have to contain at least $n+2$ nodes, contradicting with the number of node in graph. Hence it must exists such node.

And following is the algorithm to find such node.

First, run BFS on graph,then do DFS on the result. Then find whose neiborhood(the number of nodes in same layer) equals 0. These node(s) can be such node(s). ■

Problem 6

Assume you are a kind grandparent and going to give your grandchildren some pieces of cake. However, you cannot satisfy a child unless the size of the piece he receives is no less than his expected cake size. Different children may have different expected sizes. Meanwhile, you cannot give each child more than one piece. For example, if the children' s expected sizes are $[1, 3, 4]$ and you have two pieces of cake with sizes $[1, 2]$, then you could only make one child satisfied. Given the children' s expected sizes and the sizes of the cake pieces that you have, how can you make the most children satisfied? Prove that your algorithm is correct.

Solution. We can use greedy algorithm to solve this problem.

We satisfy the least expected size first to sure that do my best to hand out more cake. First, sort expected sizes$[e_1, e_2, \cdots, e_n]$ and having sizes$[h_1, h_2, \cdots, h_m]$. Then, take h_i, if $e_j \leq h_i$, give h_i to e_j ,else $i = i + 1$.($i = j = 0$ Initially) Again and again, until $i = m$. Finally done my best to hand out more cakes.

Suppose d is the optimal number of satisfying child(ren) by other algorithm. And d_0 is greedy one. We can say that $m - d_0$ cake(s) cannot satisfy any of $n - d$ child(ren). Because These cake(s) cannot

satisfy the least expected size. And less expected sizes can be satisfied by less having size cakes. Hence $d_0 = d$ always can be found. So this algorithm is correct. ■

Problem 7

Describe an efficient algorithm that, given a set $\{x_1, x_2, \cdots, x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contain all of the points. Prove that your algorithm is correct.

Solution. Greedy algorithm is also useful in this problem.

First we sort these points$\{x_1, x_2, \cdots, x_n\}$ to $\{x'_1, x'_2, \cdots, x'_n\}(x'_i \leq x_{i+1})$. Then choose $[x'_1, x'_1 + 1]$ as first unit-length closed intrevals. Next, choose the first x'_i bigger than $x'_1 + 1$ point as the next start of unit-length closed intrevals. And $x'_i + 1$ as the end. Again and again till finish all n points. Finally, get the smallest set satisfying requirement of this problem.

Next is the provement. Suppose another algorithm get a correct answer which just the ith intervals is different. And its start points is less than mine. So, the number of points in this unit-length closed intrevals must include more points. Hence my algorithm is correct. ■

Problem 8

Suppose we have an undirected graph $G = (V, E)$ with costs $c_e \geq 0$ on the edges $e \in E$. All edge costs are distinct. Assume you are given a minimum-cost spanning tree T in G. Now assume that a new edge is added to G, connecting two nodes $v, w \in V$with cost c. Give an efficient algorithm in $O(|V|)$ time to test if T remains the minimum-cost spanning tree. Please note any assumptions you make about what data structure is used to represent the tree T and the graph G.

Solution. Assump that use the union-find data structure.

First do find operation to sure if s and t are in different set. If so, T cannot remain the minimum-cost spanning tree. Or not, do DFS on tree to find if there exists a edge u which satisfying $c_u > c$ in $v - w$ path If so, T cannot remain the minimum-cost spanning tree. Or not T can remain. And the worst situation, $O(f(n)) = O(n + n - 1) = O(2n - 1) = O(n) = O(|V|)$. ■