# 20T2 真题

Quiz navigation

1 2 3 4

Finish attempt ...

Time left **2:58:34**

Question **1**

Not yet answered

Marked out of 25.00

Flag question

1. Let A and B be two sequences of numbers such that A*A = B where * denotes the convolution of sequences.

(a) Let the length of sequence A be equal to a and the length of sequence B be equal to b. Express a in terms of b. (5 pts)
(b) Find ALL sequences A such that A*A = (4,4,-3,-2,1). (20 pts)

Question **2**

Not yet answered

Marked out of 25.00

Flag question

You are late with n assignments a(1), a(2),..., a(n) that were all due today. Assignment a(i) accrues a penalty of p(i) points per day and takes t(i) days to finish. At any moment, you can work on one assignment only. Determine the order in which you should work on your assignments in order to minimise the total number of points lost. Justify the correctness of your algorithm. Your algorithm should run in time O(n log n) and you should explain why your algorithm runs in time O(n log n). (25 pts)

You are late with n assignments a(1), a(2),..., a(n) that were due today. Assignment a(i) accures a penalty of p(i) points per day and takes t(i) days to finish. At any moment, you can work on one assignment only. Determine the order in which you should work on your assignments in order to minimise the total number of points lost, justify the correctness of your algorithm. Your algorithm should run in time O(nlogn) and you should explain why.

### Solution:

get array of p/t - O(n)

sort p/t decreasing - O(nlogn)

Finish in this order - O(n)

total complexity - O(nlogn)

Prof:

Total number of points

$T = p(1)t(1) + p(2)(t(1)+t(2)) + \ldots p(k)(t(1)+\ldots t(k)) + p(k+1)(t(1) + \ldots t(k+1)) + .. + p(n)(t(1)+t(2)+\ldots+t(n))$

Assume there is a better algorithm but it breaks our greedy method.

$T' = p(1)t(1) + p(2)(t(1)+t(2)) + \ldots p(k+1)(t(1)+\ldots t(k-1)+t(k+1)) + p(k)(t(1) + \ldots t(k+1)) + .. + p(n)(t(1)+t(2)+\ldots+t(n))$

$T - T' = p(k)(t(1)+\ldots t(k)) + p(k+1)(t(1) + \ldots t(k+1)) - (p(k+1)(t(1)+\ldots t(k-1)+t(k+1)) + p(k)(t(1) + \ldots t(k+1)))$

because we know $p(k+1)/t(k+1) < p(k)/ t(k)$, $p(k+1)t(k) < p(k)t(k+1)$.

$= -p(k)t(k+1) + p(k+1)t(k) < 0$

Hence this algorithm is not better than us. => our solution is optimal.

Solve the following problem using Dynamic Programming. You are travelling along the Elbonian coast with cities c(0), c(1), c(2),..., c(n) on the shore, in that order. You are starting in city c(0) where a famous spa is, and need to reach the airport situated in city c(n); thus, you will be going through all cities c(0), c(1),..., c(n-1), c(n) in that order. In each city you must swap the animal you are riding on and the choices are a camel, a horse, a mule and a donkey, denoted C,H,M,D respectively. However, each city has its own rules what kind of animal exchanges are allowed. For example, in some of the cities you can swap a horse only for a donkey or a mule, in some of the cities you can swap a camel only for another camel or a horse, and so on. You know all the rules of all the cities c(1),...,c(n), expressed by a function R(i,a,b) given by R(i,a,b)=1 if in city c(i) one can swap animal a for animal b and zero otherwise (a and b belong to the set {C,H,M,D}). You also know the speed v(a) (a = C,H,M,D) of each of the four animals, as well as the distances d(i) between cities c(i-1) and c(i) for all i = 1,2,...,n. You have to design an algorithm for computing the minimal amount of travel time needed to travel from city c(0) all the way to city c(n) as well as an animal swapping strategy which allows you to travel in such a minimal amount of time. In the starting city c(0) you can choose to start your journey riding any of these four animals. To solve this problem do the following tasks in this order:
(a) Formulate precisely the subproblems you are going to solve. (10 pts)
(b) Write the exact recursion equations. (12 pts)
(c) Explain how the solution to the original problem is obtained from the solutions of the subproblems you have defined. (2 pts)
(d) Estimate the asymptotic run time of your algorithm in terms of the number n of cities. (1 pt)

Solve the following problem using Dynamic Programming. You are travelling along the Elbonian coast with cities c(0), c(1), c(2),...,c(n) on the shore, in that order. You are starting in city c(0) where a famous spa is, and need to reach the airport situated in city c(n); thus, you will be going through all cities c(0), c(1), c(2),...,c(n) in that order. In each city you must swap the animal you are riding on and the choices are a camel, a house, a mule and a donkey, denoted C,H, M, D respectively. However, each city has its own rules what kind of animal exchanges are allowed. For example, in some cities you can swap a horse only for a donkey or a mule. In some of the cities you can swap a camel only for another camel or a horse, and so on. You know the rules of all the cities...

## Solution

**Subproblem**: Opt(a, i) 到达城市i的时候骑动物a的最短时间

**recursion**：

R(i, a, b)

Opt(C, i) = Min(opt(a, i-1) + di/vC where `a` satisfies R(i-1, a, C) = 1) 如果没有满足的a，我们就设为infinity

Opt(H, i) = Min(opt(a, i-1)+ di/vH where `a` satisfies R(i-1, a, H) = 1) 如果没有满足的a，我们就设为infinity

opt(M, i) = Min(opt(a, i-1) + di/vM where `a` satisfies R(i-1, a, M) = 1) 如果没有满足的a，我们就设为infinity

opt(D, i) = Min(opt(a, i-1) + di/vD where `a` satisfies R(i-1, a, D) = 1) 如果没有满足的a，我们就设为infinity

**Final Solution**:

Min(opt(a, n), a 属于{C,H,M,D})

**Time Complexity:**

O(n). For each city we only iterate once.

Question 4
Not yet answered
Marked out of 25.00
Flag question

(a) Describe in detail how max flow algorithms can be used to find a maximum matching in bipartite graphs. (5 pts)
(b) You are given a sequence A(1),..., A(N) of N distinct positive integers and an integer M. You have to determine if it is possible to assign to each element A(i) a distinct integer B(i) larger or equal to 2 and smaller or equal than M such that for all i between 1 and N integer A(i) is divisible by integer B(i). Different A(i) must be assigned different B(i). (20 pts)

bipartite graph:

Left vertices : A (N distinct positive numbers)

right vertices: B (from 2 to M)

add super source and super sink. Set capacity to 1

connect A(i) to B(i) where A(i)|B(i), set capacity to 1

run max-flow algorithm, if the output equals the input size N, there exists such a solution. Otherwise, no.