# CS 5381

# Analysis of Algorithms

1. Select the correct answer (1 point for each question)
   1. Which of the following running time is :

a). b).

c). d). Both b) and c)

e). Both a) and b)

* 1. Consider the knapsack problem: There are items, each item has a profit and weight . There is a knapsack of size . The goal is to select a subset of items whose total weight is at most and the total profit is maximized. What is its input size?

a).

b).

c).

d).

1.3 Consider the following graph.

Chart

Description automatically generated

Which of the following edge set can belong to a spanning tree:

a).

b).

c).

d).

1.4 Consider an undirected graph with vertices and edges, every edge has length 1. There is a source vertex and a sink vertex , and the problem is to find a shortest path from to . Which of the following is true?

a). Breadth-First search algorithm can be used to find a shortest path, while Dijkstra's algorithm cannot.

b). Breadth-First search algorithm cannot be used to find a shortest path, while Dijkstra's algorithm can.

c). Both Breadth-First search algorithm and Dijkstra's algorithm can be used to find a shortest path

d). Neither Breadth-First search algorithm and Dijkstra's algorithm can be used to find a shortest path

* 1. Consider the algorithm for closest pair (in Slides Divide and Conquer I, page 74). Suppose we are given a set of points in -dimensional Euclidean space, with the distance of any two points given by the standard -norm, i.e., . We want to find two points whose distance is minimal, which of the following statement is true:

a). The closest pair algorithm does not work in Euclidean space

b). The closest pair algorithm works for , but not for .

c). The closest pair algorithm works, but its running time is exponential in , and is thus only in polynomial time when is a constant

d). The closest pair algorithm works and runs in polynomial time for any input

1.6 There are teams, ranked from 1 to . Alice can see the ranking, but Bob cannot. To figure out the ranking, Bob can make queries to Alice. In each query, Bob can choose two teams and , and say " ranks higher than ", and Alice will reply "Yes" or "No" based on the ranking she sees. Alice always replies truthfully. Which of the following is true:

a). Bob can make queries according to merge sort, and it takes queries for him to discover the whole ranking in the worst case

b). Bob can make queries according to randomized quick sort, and it takes queries for him to discover the whole ranking in the worst case

c). Bob cannot use merge sort nor quick sort, because these are algorithms for sorting numbers, which do not apply in ranking teams

d). Bob can make queries according to merge sort but not randomized quick sort, because a team cannot be chosen as a pivot

1.7 Consider the interval scheduling problem (Dynamic programming I, page 4): there are jobs, each job has a start time, a finish time, and a weight. Two jobs are compatible if they do not overlap. The goal is to find max-weight subset of mutually compatible jobs. Which statement is true:

a). The dynamic programming algorithm at page 14 does not work if the start times and finish times of jobs are fractional. It only works when they are integers.

b). The dynamic programming algorithm at page 14 does not work if the weights of jobs are fractional. It only works when they are integers.

c). If the weight of every job is 1 and the start times and end times are integers, then the greedy algorithm (earliest finish time first at page 12, Greedy I) works. However, if the start times and end times are fractional, then the greedy algorithm does not work.

d). None of the above

1.8 Consider the multiplication of an -bit integer with an -bit integer. Which of the following statement is true?

a). FFT algorithm requires bit operations, where

b). FFT algorithm requires bit operations, where

c). FFT algorithm requires bit operations, where

d). FFT algorithm does not work because we cannot transform the multiplication of -bit integer and -bit integer to the multiplication of two integers of the same bit number

1.9 Consider the sequence alignment problem (page 3, Dynamic programming II). Assuming gap penalty = 2 and mismatch penalty = 1, what is the edit distance between the string **PALETTE** and the string **PALATE**

a).

b).

c).

d).

1.10 Consider segmented least square problem (page 26, dynamic programming I). What if the constant integer in the objective is very huge (approaching ), and what if is very small (approaching 1). Select the **correct** answer:

a). No matter is large or small, the textbook algorithm (page 28) always works and runs polynomially in the input size

b). The textbook algorithm (page 28) only works when is small. If is very large, the algorithm cannot guarantee to find the correct solution

c). The textbook algorithm (page 28) only works when is large. If is very small, the algorithm cannot guarantee to find the correct solution

d). No matter is large or small, the textbook algorithm (page 28) always works. But if is very huge, then the running time becomes pseudo-polynomial (i.e., polynomial in ) instead of polynomial in

In the following questions, denotes the natural numbers. By , we mean is a natural number.

1. (2 points) [Coin exchange] Given are 9 coin denominations and a target value .

2.1 Give the solution returned by Cashier's algorithm.

2.1 Find out the best solution.

1. (6 points) [Knapsack] Given are items, each item is associated with a value and a weight . The goal is to find a subset of items whose total weight is at most , and whose total profit is maximized.

3.1 (1 point) Consider the following list of items. Find out a subset of items whose total profit is exactly **50**, and whose total weight is minimized.

|  |  |  |
| --- | --- | --- |
| **items** | **value** | **weight** |
| 1 | 1 | 1 |
| 2 | 6 | 2 |
| 3 | 18 | 5 |
| 4 | 22 | 6 |
| 5 | 28 | 7 |

3.2 (2 points) Design a dynamic programming algorithm for the knapsack problem of running time , where

3.3 [2dimensional knapsack] Consider the following generalization of the knapsack problem. Given are items, each item is associated a weight . There are two people, Alice and Bob, who have different view on the values of items. Particularly, item has a value to Alice, and a value to Bob. The quality of a solution is measured by the least happiest person among Alice and Bob, that is, a subset of items will have a total value to Alice and total value to Bob, and we define as the quality of the solution. See the following table. If we choose item 1 and 2, then their total weight is 3. Their total value to Alice is 50, and total value to Bob is 7. The quality of this solution is 7.

|  |  |  |  |
| --- | --- | --- | --- |
| **items** | **value to Alice** | **value to Bob** | **weight** |
| 1 | 28 | 1 | 1 |
| 2 | 22 | 6 | 2 |
| 3 | 18 | 18 | 5 |
| 4 | 6 | 22 | 2 |
| 5 | 1 | 28 | 1 |

(1 points) Given a knapsack budget , find out the solution with the largest quality.

3.4 (2 points) Design a dynamic programming algorithm to find out a subset of items whose total weight is at most and whose quality is maximized.

1. (7 points) [Quality of Service (QoS)] Given are jobs, each job has a processing time , and a weight . There is one machine, every job has to be processed non-preemptively on this machine. A schedule is a permutation of all jobs. The completion time of a job is the time when this job finishes. The quality of service is measured by the weighted completion time . For example:

|  |  |  |
| --- | --- | --- |
| **jobs** | **processing time** | **weight** |
| 1 | 1 | 5 |
| 2 | 2 | 4 |
| 3 | 3 | 3 |
| 4 | 4 | 4 |
| 5 | 5 | 5 |

Any permutation, say, (2,3,4,1,5), can be a feasible schedule. In this case, the first job is job 2, finishes at time . The second job is job 3, finishes at time . Similarly , , . The weighted completion time is given by

4.1(1 point) Suppose there are only two jobs, . Compute the quality of two permutations (schedules) (1,2) and (2,1). Which schedule has a smaller value?

4.2 (1 points) Suppose there are only two jobs, such that . Compare the quality of two permutations (schedules) (1,2) and (2,1). Which schedule has a smaller value?

4.3 (1 point) Back the the example with 5 jobs. Compute and compare the two permutations (2,3,4,1,5) and (2,3,4,5,1). Which one has a smaller value?

|  |  |  |
| --- | --- | --- |
| **jobs** | **processing time** | **weight** |
| 1 | 1 | 5 |
| 2 | 2 | 4 |
| 3 | 3 | 3 |
| 4 | 4 | 4 |
| 5 | 5 | 5 |

4.4 (2 points) Consider the general problem. Suppose two jobs satisfies that . Consider the two permutations and such that and are identical except for job 1 and 2, i.e., we can transform to by swapping job 1 and job 2. Compare the quality of two permutations, which one has the smaller value?

4.5. (2 points) Based on the above questions, suppose , then what is the solution with the smallest weighted completion time?

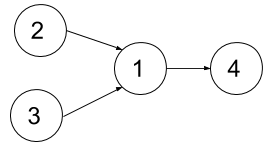
1. (5 points) [Parallel courses]

Given the integer n representing the number of courses at some university labeled from 1 to n, and the array dependencies where dependencies[i] = [,] represents a prerequisite relationship, that is, the course  must be taken before the course . Also, you are given the integer k.

In one semester you can take at most k courses as long as you have taken all the prerequisites for the courses you are taking.

*Return the minimum number of semesters to take all courses*. It is guaranteed that you can take all courses in some way.

**Example 1:**



**Input:** n = 4, dependencies = [[2,1],[3,1],[1,4]], k = 2

**Output:** 3

**Explanation:** The figure above represents the given graph. In this case we can take courses 2 and 3 in the first semester, then take course 1 in the second semester and finally take course 4 in the third semester.

**Example 2:**

A picture containing text, pool ball

Description automatically generated

**Input:** n = 5, dependencies = [[2,1],[3,1],[4,1],[1,5]], k = 2

**Output:** 4

**Explanation:** The figure above represents the given graph. In this case one optimal way to take all courses is: take courses 2 and 3 in the first semester and take course 4 in the second semester, then take course 1 in the third semester and finally take course 5 in the fourth semester.

**Example 3:**

**Input:** n = 11, dependencies = [], k = 2

**Output:** 6

5.1 (2 points) Design an arbitrary greedy algorithm. Your greedy algorithm should apply to any given instance. Give an example instance when your greedy algorithm finds the optimal solution (you can use the examples above).

5.2. (3 points) In general, finding the minimal number of semesters for an arbitrary dependency graph is NP-hard, which means that a polynomial time algorithm is unlikely to exist. But an algorithm running in exponential time can be designed using dynamic programming. Design an algorithm of running time where is the number of courses.

(Hint: consider the subset of courses taken in semesters.)