**CS4335 Design and Analysis of Algorithms (Midterm, 2021)**

**If you think some questions are ambiguous, you may write your assumptions clearly. However, you assumption should not trivialize the questions.**

**Question 1. (20 points)**

**(a)** (10 points) For the interval scheduling problem, the set of jobs (si, fi) are as follows:

(0, 2), (1, 3), (2, 6), (2, 4), (6, 9) (8, 12), (5, 8), and (6, 7).

Use a greedy algorithm to compute the maximum number of compatible jobs. You should give main steps. What is the running time of the greedy algorithm?

**Answer:**

Sorting based on finish time: (0, 2), (3, 4), (1, 5), (2, 6), (5, 7), (7, 10), (8, 12) (9,13)

Choose

{(0, 2)}

{(0, 2), (3, 4)} (1, 5) (2, 6) overlap

{(0, 2), (3, 4), (5, 7)}

{(0, 2), (3, 4), (5, 7), (7, 10)} (8, 12) (9, 13) overlap

Max = 4 jobs

Running time: O(nlogn)

**(b)** (8 points) For the interval partitioning problem, the set of lectures (si, fi) are as follows:

(0, 1), (0, 3), (1, 4), (2, 6), (2, 4), (4, 5), (3, 5) and (5, 8).

Use a greedy algorithm to compute the minimum number of classrooms to accommodate all the lectures. You should give main steps.

**Answer:**

Sort based on start time: (0, 2), (0, 4), (2, 5), (3, 5), (3, 6), (4, 7), (5, 8), (6, 9)

Room 1: (0, 2), (2, 5), (5, 8)

Room 2: (0, 4), (4, 7)

Room 3: (3, 5), (6, 9)

Room 4: (3, 6)

**(c)** (2 points) For the interval partitioning problem given in (b), what is the depth of the problem?

**Answer:**

Depth = 4

**Question 2. (20 points)**

1. **(7 points)** Find the minimum spanning tree for the graph in Figure 1 using Kruskal’s algorithm.

**Answer:**

First, we sort the edges in ascending order of their weights:

,

Then, we select edges of which the two vertices are in different trees: .

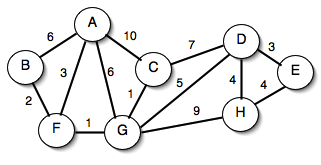


Figure 1

1. **(8 points)** Find the minimum spanning tree for the graph in Figure 1 using Prim’s algorithm.

**Answer:**

Solution 1:

Define: , ,

Steps:

1. , ,

2. , ,

3. , ,

4. , ,

5. , ,

6. , ,

7. , ,

The minimum spanning tree by Prim is .

Solution 2:

Start from A.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H |
| Start | 0/NIL | INF/NIL | INF/NIL | INF/NIL | INF/NIL | INF/NIL | INF/NIL | INF/NIL |
| Pick A |  | 6/A | 10/A | INF/NIL | INF/NIL | 3/A | 6/A | INF/NIL |
| Pick F |  | 2/F | 10/A | INF/NIL | INF/NIL |  | 1/F | INF/NIL |
| Pick G |  | 2/F | 1/G | 5/G | INF/NIL |  |  | 9/G |
| Pick C |  | 2/F |  | 5/G | INF/NIL |  |  | 9/G |
| Pick B |  |  |  | 5/G | INF/NIL |  |  | 9/G |
| Pick D |  |  |  |  | 3/D |  |  | 4/D |
| Pick E |  |  |  |  |  |  |  | 4/D |
| Pick H |  |  |  |  |  |  |  |  |

**Edgeset={BF, CG, DG, ED, FA, GF, HD }**

1. (**5 points**) Is the path between a pair of vertices in a minimum spanning tree of an undirected graph necessarily a shortest path? Justify your answer.

**Answer:**

No, a path between a pair of vertices in a minimum spanning tree is not necessarily their shortest path. For example, the shortest path between H and E should be H->E, of which the distance is 4. However, in an MST, E can only be reached via D from H, of which the total distance is 7, and is longer than the shortest path.

**Question 3. (15 points)**

Use Dijkstra’s algorithm to compute a shortest path from *a* to *i* in the following graph. You should give main steps.

A picture containing watch, different, arranged

Description automatically generated

**Answer:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| iteration | A | b | c | d | e | f | g | h | i |
| 0 | 0/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil |
| 1 |  | 4/a | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil |
| 2 |  |  | 15/b | 13/b | ∞/nil | ∞/nil | ∞/nil | ∞/nil | ∞/nil |
| 3 |  |  | 15/b |  | 15/d | 19/d | ∞/nil | ∞/nil | ∞/nil |
| 4 |  |  | 15/b |  |  | 19/d | 22/e | 19/e | ∞/nil |
| 5 |  |  |  |  |  | 19/d | 22/e | 19/e | ∞/nil |
| 6 |  |  |  |  |  | 19/d | 22/e |  | 29/h |
|  |  |  |  |  |  |  | 22/e |  | 29/h |
|  |  |  |  |  |  |  |  |  | 29/h |

The shortest path is: a-b-d-e-h-i

The shortest length is: 29

**Question 4 (15 points)**

1. (**9 points)** For the list:2, 1, 5, 8, 9, 10, 4, 7, 6, 13, 14, and 11. Suppose we have sorted the two halves as list1: 1, 2, 5, 8, 9, 10; and list2: 4, 6, 7, 11, 13, 14. Calculate the number of inversions with one number in list1 and the other number in list2 using O(n) operations. Immediate steps are required.
2. **(3 points).** Assume T(n) is the running time for the following algorithm. List the recursive relation, and with it, what is T(n) in terms of big-O?

**FindMax(A, k, n)**

**Input:** Array A of size n, and an integer k<n

**Output:** the maximum element from A[k], A[k+1], …, A[n-1]

**if** k<n-1

**return** max(A[k], FindMax(A, k+1, n))

**else return** A[k]

**Initial call FindMax(A, 0, n)**

1. (**1 point**) Suppose *T(1)=1,* and *T(n)=T(n-1)+n*. What is *T(n)* in terms of big O notation?
2. **(2 points)** Suppose *T(1)=1,* and *T(n)=T(n/3)+1*. What is *T(n)* in terms of big O notation?

**Answer:**

1, 2, 5, 8, 9, 10 4, 6, 7, 11, 13, 14

4 : 4

6 : 3

7 : 3

11 : 0

13 : 0

14 : 0

Sum of inversions : 10

(b) The recurrence expression for is : . Hence, .

(c) , hence .

(d) . Assume, , we have .

In this way, =1+log n/. Hence, .

**Question 5. (15 points)**

Given an array of n ≥ 2 **distinct** integers (i.e., no two integers are the same) sorted in ascending order, say [x(1),...,x(n)], we want to find the absolute minimum *difference between the x(i) and i*. For example, for *x = [-10, 9, 10, 12, 13, 16] ,* the minimum *difference d =*|*x(2)−2| =* |*9−2| = 7*.

(a) (**5 points)** Use a linear time algorithm to solve the problem.

(b) (**5 points**) Use a divide and conquer approach the solve the problem. The running time should be O(logn).

**Answer:**

(c) **(5 points)** Set up and solve a recurrence equation for part (b) to estimate the running time of your algorithm. Prove that the running time of your algorithm is O(logn).

**Hint:**

1. The difference will first decrease and then increase.

**Answer:**

**a)**

min ←∞

**for** i ←1 to n:

**if** x[i]-i < min

min← |x[i]-i|

**return** min

**b)**

minDiff(*l*, r)

**if** r-l=0 or r-*l*=1

**return** min(|x[*l*]-*l*|, |x[r]-r|)

mid= ⎣(l+r)/2⎦

**if** x[mid]>mid:

**return** minDiff(*l*, mid)

**else**

**return** minDiff(mid, r)

initial call minDiff(1, n)

**c)**

T(n)=T(n/2) +1

T(n/2)=T(n/22) +1

…

T(n/2k-1))=T(n/2k) +1

Then T(n)= T(n/2k) +k

Assume n/2k=1, and we have k=log2n.

T(n)=1+ log2n, and T(n)=O(log n)

**Question 6.** **(15 points)**

Suppose we have an array of n positive integers. A contiguous subarray A[i .. j] is called a squared interval if the sum of its entries is a squared number. Design a greedy algorithm to compute the maximum number of squared intervals such that every entry in A will be covered at most once. You can state your algorithm in English or in Pseudo code (**5 points**). What is the running time of algorithm in big-Oh (**5 points**)? Prove that your algorithm is correct. **(5 points)**

4, squared

16, squared

36, squared

25, squared

1 7 7 5 5 3 4 7 6 9 1 13 5

**Answer 1:**

Outline of Solution to Q6:

1. Algorithm:

Phase 1:

For each i=1 to n.

Find the shortest squared interval ending at i. If no such interval, return nil.

// For each i, the running time is at most O(n), thus, the total running time for Phase 1 is O(n^2) . There are at most n intervals found by Phase 1. You should give the details of how to find the shortest squared interval.

Phase 2: Let the squared intervals obtained by phase 1 as inputs and use the interval scheduling algorithm to find the maximal number of the compatible intervals.

// the running time of Phase 2 is O (n), since the intervals from Phase 1 are sorted according to finish time, and no sorting is necessary.

1. Running time =O(n^2).
2. Outline of Proof:

(1) Let A be a maximal set of non-overlapped squared interval. If A contains an interval starting at i, replace this interval by the shortest squared interval starting at i will not change the optimality of A. (2) For each i, A can contain at most one interval starting at i.

(1)+(2) indicate that this problem is equivalent to the interval scheduling problem and the algorithm is correct.