CSPB 3104 - Park - Algorithms

Dashboard / My courses / 2241:CSPB 3104 / 6 May - 12 May / Final Exam (Remotely Proctored)

Started on	Tuesday, 7 May 2024, 12:44 PM
State	Finished
Completed on	Tuesday, 7 May 2024, 2:33 PM
Time taken	1 hour 49 mins
Grade	Not yet graded

Question **1**

Correct

Mark 1.00 out of 1.00

Let G be a connected, undirected graph with 42 nodes and 600 edges. How many edges does minimum spanning tree in G have?

Answer: 41 ✓

The correct answer is: 41

Question **2**

Correct

Mark 1.00 out of 1.00

Let H be an open address hashtable with 3104 slots. What is the maximum number of keys that can be inserted into H?

Answer: 3104 ✓

The correct answer is: 3104

Question **3**

Correct

Mark 1.00 out of 1.00

What is the largest number of (Maximal) Strongly Connected Components in a graph with n = 375 nodes and m = 637 edges?

Answer: 375 ✓

The correct answer is: 375

Question **4**

Correct

Mark 1.00 out of 1.00

What is the smallest number of (Maximal) Strongly Connected Components in a graph with n = 30 nodes and m = 30 edges?

Answer: 1 ✓

The correct answer is: 1

Question **5**

Correct

Mark 2.00 out of 2.00

What is the worst-case complexity of inserting an element into a red-black tree with n nodes?

Answer: log(n) ✓

The correct answer is: log(n)

Question **6**

Correct

Mark 2.00 out of 2.00

Let G be an undirected graph with n nodes. We treat each undirected edge $\{u, v\}$ as two directed edges (u, v) and (v, u). Which types of edges will not be seen during a depth-first traversal (DFS) of G? (Choose between Tree, Back, Forward or Cross Edge).

Answer: Cross Edge ✓

The correct answer is: Cross

Question **7**

Correct

Mark 2.00 out of 2.00

Let G be a directed acyclic graph (DAG) with n nodes. Which types of edges will not be seen during a depth-first traversal (DFS) of G? (Choose between Tree, Back, Forward or Cross Edge).

Answer: Back Edge ✓

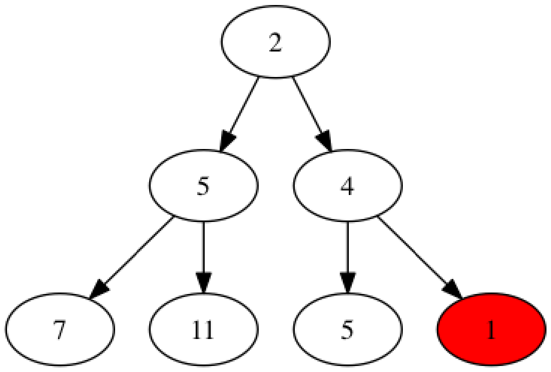
The correct answer is: Back

Question **8**

Correct

Mark 2.00 out of 2.00

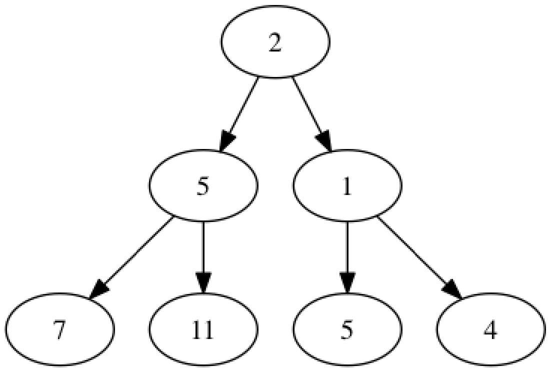
Consider the heap below where node labelled "1" is shown in red has just been inserted.



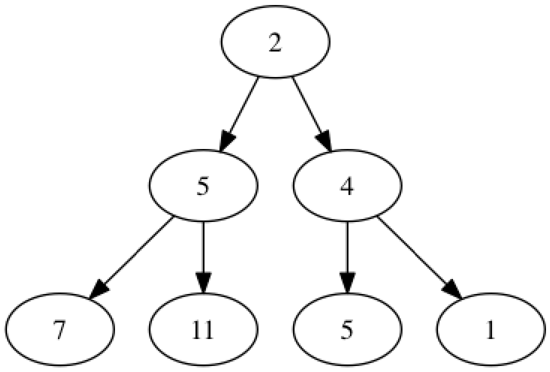
Choose the final heap that is obtained after the newly inserted node is "bubbled up" to its final position.

Select one:

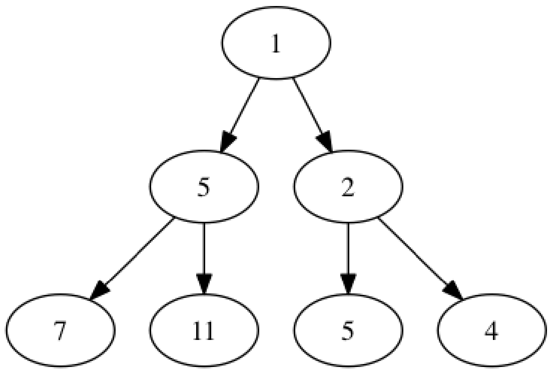
a.



b.

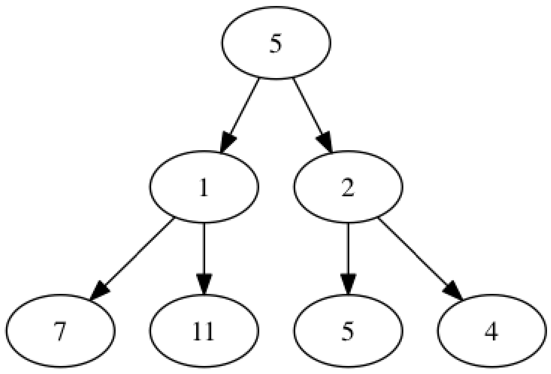


[✓ c.]



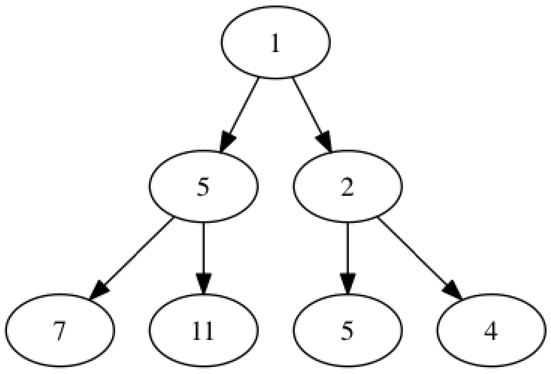
✓

d.



Your answer is correct.

The correct answer is:



Question **9**

Correct

Mark 2.00 out of 2.00

Consider the function below:

```
def foo(a):
    n = len(a)
    total = 1
    count = n
    while (count >= 0):
        count = count - 1
        for i in range(count):
            if a[i] == 0:
                return 0
            else:
                total = (total + count) * a[i]
    return total
```

Select all true statements from the list below. Let n be the size of the input array a.

Select one or more:

- [✓ a. The best case running time is $\Theta(1)$] ✓
- b. The worst case running time is $\Theta(nlog(n))$
- c. The best case running time is $\Theta(n)$
- d. The worst case running time is $\Theta(1)$
- e. The worst case is realized on an array whose elements are all 0s.
- [✓ f. The worst case running time is $\Theta(n^2)$] ✓
- [✓ g. The worst case is realized on an array which does not contain a 0.] ✓

Your answer is correct.

The correct answers are: The worst case running time is $\Theta(n^2)$
, The best case running time is $\Theta(1)$
, The worst case is realized on an array which does not contain a 0.

Question **10**

Correct

Mark 2.00 out of 2.00

Given an array a of size n and value v, we wish to find the element $a[j]$ that is "closest" to v. Mathematically, the closest element is one that minimizes $|a[j] - v|$, the absolute value of the difference between the element and v.

If there are multiple elements in the array that are equally close then any of the elements can be returned.

What is the complexity of the best algorithm to solve this problem? Choose all valid options from those given below.

Select one or more:

- ☐ a. The best approach is to use merge sort to sort the array in ascending order and then perform a modified binary search algorithm.
- ☑ b. If the array a is sorted, then the worst case complexity is $\Theta(\log(n))$ obtained by modifying binary search. ✓
- ☐ c. We can solve the problem in $\Theta(1)$ time by simply checking if the middle element of the array equals v or not.
- ☑ d. If the array a is unsorted, then the worst case complexity is $\Theta(n)$ obtained by scanning the elements of the array one by one. ✓
- ☐ e. The best approach is to use insertion sort to sort the array in ascending order and then perform a scan of the array a from left to right.

Your answer is correct.

The correct answers are: If the array a is unsorted, then the worst case complexity is $\Theta(n)$ obtained by scanning the elements of the array one by one.
, If the array a is sorted, then the worst case complexity is $\Theta(\log(n))$ obtained by modifying binary search.

Question **11**

Correct

Mark 3.00 out of 3.00

Consider the following list of functions:

$f_1 : \log(n),\ f_2 : n\log(n),\ f_3 : n^2,\ f_4 :\ n^{1.2},\ f_5 :\ n^2\log(n),\ f_6 : 1.2$

Arrange the functions in asymptotically increasing order from asymptotically smallest to largest. Use the boxes below to write down the IDs of the functions from smallest first to the largest last. Just fill in a number: for instance, write "6" to indicate f_6.

6 ✓

1 ✓

2 ✓

4 ✓

3 ✓

5 ✓

Question **12**

Partially correct

Mark 0.50 out of 3.00

Consider a graph with 6 nodes and 8 edges and edge weights non negative.

Suppose we wish to find single source shortest path from a given node in the graph.

Select all the choices that are true.

Select one or more:

- [ ] a. Dijkstra algorithm cannot be run for such a graph.
- [✓] b. Dijkstra algorithm will perform 6 edge relaxations ✗
- [✓] c. Bellman-Ford will perform 48 relaxations. ✗
- [✓] d. The graph cannot have negative weight cycles ✓
- [ ] e. Dijkstra algorithm will perform 8 edge relaxations.

Your answer is partially correct.

The correct answers are: Dijkstra algorithm will perform 8 edge relaxations., The graph cannot have negative weight cycles

Question **13**

Partially correct

Mark 1.65 out of 3.00

Let $G = (V, E)$ be a weighted graph with n vertices and m edges. Select all of the following which is true.

Select one or more:

- [ ] a. It is possible to find the longest path between two vertices in G by negating all the edge weights and running any shortest path algorithm.
- [✓] b. If G is an undirected graph, then it is impossible to find the shortest cycle in G. ✗
- [✓] c. If G is connected and $m = n - 1$ then G is acyclic. ✓
- [✓] d. It is possible to detect the existence of a cycle in G in $O(n + m)$ time. ✓
- [✓] e. If G contains a negative weight cycle, then the shortest path between any pair of vertices in G is undefined. ✓
- [✓] f. If G is a DAG, then it is possible to compute the longest path between two vertices in polynomial time. ✓
- [ ] g. If G is a strongly connected DAG then G must contain a directed cycle.
- [ ] h.
  If G is undirected and $m \geq n$ then G must contain a cycle.

Your answer is partially correct.

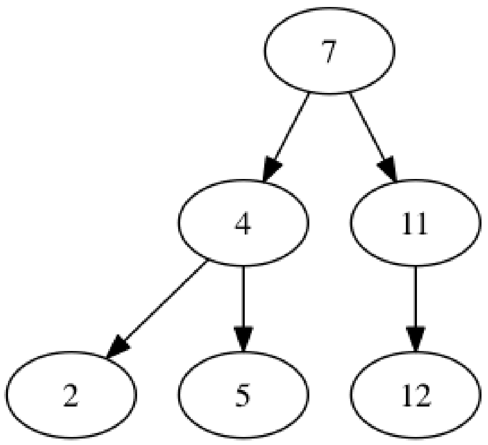
You have correctly selected 4.
The correct answers are: It is possible to detect the existence of a cycle in G in $O(n + m)$ time.
, If G is connected and $m = n - 1$ then G is acyclic.
,
If G is undirected and $m \geq n$ then G must contain a cycle.
, If G is a DAG, then it is possible to compute the longest path between two vertices in polynomial time.
, If G contains a negative weight cycle, then the shortest path between any pair of vertices in G is undefined.

Question **14**

Correct

Mark 3.00 out of 3.00

Consider the tree below:



Suppose we wish to right rotate the tree with $x = 4$ and $y = 7$, so that the new root is 4.

Write down the numbers corresponding to the nodes below in the tree after rotation:

The right child of the new root (4) will be:

7 ✓

The left child of 7 in the new tree will be:

5 ✓

Which of the following remain the same during the right rotation?

- [✓ Left subtree of 4 and right subtree of 7] ✓ Correct
- Right subtree of 4 and right subtree of 7
- Left subtree of 4 and left subtree of 7
- Right subtree of 4 and left subtree of 7

Mark 1.00 out of 1.00

The correct answer is: Left subtree of 4 and right subtree of 7

Question **15**

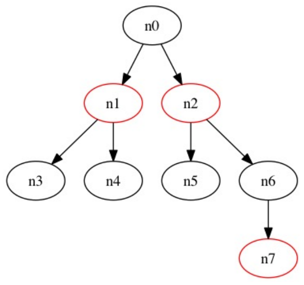
Correct

Mark 4.00 out of 4.00

This problem involves the consequences of the four basic rules of a red-black tree, recalled below:

1. Every node must be red or black.
2. The root must be black and the leaves must be labelled nil, and colored black
3. Both the children of a red node are black.
4. For any node n, all paths from n to its descendent leaves must encounter the same number of black nodes.

Consider tree T1 shown below



The leaves "nil" are not shown in the diagram.

(A) The tree above is a valid red-black tree satisfying the properties listed above? Write TRUE/FALSE in the space below.

TRUE ✓

(B) What is the height of the tree? Enter the number in the space below.

4 ✓

(C) What is the black height of the root **n0** of the tree? Enter the number in the space below.

2 ✓

(D) Which of the following changes to the tree will **NOT** invalidate the red-black properties?

- [✓ Changing nodes n1 and n2 to both be black] ✓ Correct
- Changing the color of node n7 to black
- Changing the color of n5 to red
- Making all nodes in the tree black
- None of the changes can be made in the list can be made.

Mark 1.00 out of 1.00

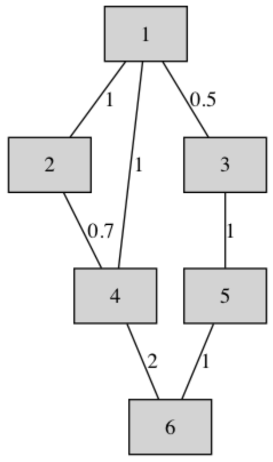
The correct answer is: Changing nodes n1 and n2 to both be black

Question **16**

Correct

Mark 5.00 out of 5.00

Consider the following graph below:



We wish to calculate a Minimal Spanning Tree (MST) for the graph. Answer the questions below concerning MSTs.

(A) Which of the following edges **must** be part of every MST for this graph?

- (4,6)
- (1,4)
- (1,2)
- [✓ (1,3)] ✔ Correct

Mark 2.00 out of 2.00

The correct answer is: (1,3)

(B) Which of the following edges can never be part of an MST for this graph?

- (1,3)
- [✓ (4,6)] ✔ Correct
- (1,2)
- (1,4)

Mark 2.00 out of 2.00

The correct answer is: (4,6)

(C) If we were to use Kruskal's algorithm to construct an MST, what is the first edge to be added to this MST?

- [✓ (1,3)] ✔ Correct
- (1,2)
- (4,6)
- (1,4)

Mark 2.00 out of 2.00

The correct answer is: (1,3)

(D) If we were to use Prim's algorithm to construct an MST starting with vertex 2, what is the first edge to be added to this MST?

- (1,3)
- [✓ (2,4)] ✓ Correct
- (2,1)
- (1,4)

Mark 2.00 out of 2.00

The correct answer is: (2,4)

(F) What is the total sum of edge weights for the MST?

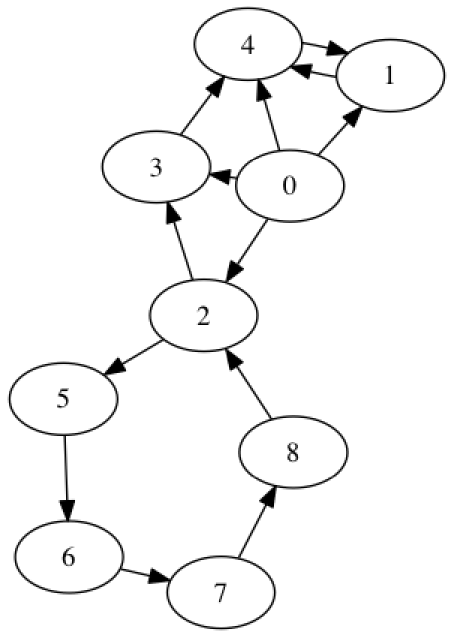
4.2 ✓

Question **17**

Correct

Mark 5.00 out of 5.00

Consider the graph shown below with 9 nodes:



Which of the sets below is a strongly connected component of the graph?

- ○ 0,2,3
- ○ 0,2,3,5,6,7,8
- ○ All the nodes in the graph
- [✓ 2,5,6,7,8] ✓ Correct
- ○ 2

Mark 2.00 out of 2.00

The correct answer is: 2,5,6,7,8

How many strongly connected components are there in the graph?

4 ✓

Suppose a DFS is called on the graph G starting from node 0, choose the node with the largest finish time?

Assume that the DFS chooses to visit adjacent vertices of a node in the increasing order of their IDs.

0 ✓

Let G^R be the graph with the edges reversed. What are the nodes visited when a DFS of G^R is performed starting from the node 0?

Write down the sequence of nodes visited in G^R separated by commas (,). Please avoid whitespaces in your answer.

0 ✓

After calling dfsVisit on G^R starting from nodes 0, suppose we were to call dfsVisit on node 2 for the graph G^R. Write down the sequence of nodes visited by this call?In particular, nodes visited in the previous problem will not be revisited.

2,8,7,6,5 ✓

Question **18**

Partially correct

Mark 4.00 out of 6.00

We have a list of integers $lst[0], lst[1], \cdots, lst[n-1]$ and a lower limit target L.

- We wish to select the least number of elements from lst so that their sum is **greater than or equal to** L.
- As a special case, we are allowed to choose 0 (or no elements) from lst and the sum would be 0. This would work whenever $L \leq 0$, for instance.

Examples:

list is [10, 11, 12, 13, 16, 19, 27]

L = 25

The optimal answer is to choose just one number [27] from the list to make up a sum of 27 *which is more than* 25.

Same list but with L = -15

The optimal answer is to choose nothing (or 0 numbers) from the list and make up a sum of 0 *which is more than* -15.

Let **minSubset(i, S)** denote a recurrence that computes the minimum number of integers to be chosen from the sublist $lst[0], \ldots, lst[i-1]$ of the first i elements of the list to achieve a target sum **greater than or equal to** S.

We wish to write a recurrence for **minSubset(i, S)**

Select the correct choice for the base case for **minSubset(i, S) when S < 0?**

[✓ 0] ✓ Correct

INFINITY

-INFINITY

1

Mark 1.00 out of 1.00

The correct answer is: 0

Select all the correct choices for the base case for **minSubset(i, S) when i = 0, S > 0?**

0

[✓ INFINITY] ✓ Correct

-INFINITY

1

Mark 1.00 out of 1.00

The correct answer is: INFINITY

Select all the correct choices for the base case for **minSubset(i, S) when S = 0?**

[✓ 0] ✓ Correct, because you are asked to make a sum of 0, that can be done with 0 elements no matter what integers are given to us

INFINITY

-INFINITY

○ 1

Mark 1.00 out of 1.00

The correct answer is: 0

Consider the recurrence relation for **minSubset(i, S)** partially shown below:

$$minSubset(i,S) : \min \begin{pmatrix} ??_1 + minSubset(i-1, S{-}??_2) \\ ??_3 + minSubset(i-1, S) \end{pmatrix}.$$

Select all the correct choices for the missing fields shown by $??_1$.

[✓ 0] ✖ Incorrect

○ INFINITY

○ 1

Mark 0.00 out of 1.00

The correct answer is: 1

Select all the correct choices for the missing fields shown by $??_2$.

○ lst[i]

[✓ lst[i-1]] ✔ Correct: if we are allowed to use the first i elements, then lst[i-1] is the last element

○ 0

Mark 1.00 out of 1.00

The correct answer is: lst[i-1]

Select all the correct choices for the missing fields shown by $??_3$.

○ 0

○ INFINITY

[✓ 1] ✖ Incorrect

Mark 0.00 out of 1.00

The correct answer is: 0

Question **19**

Complete

Marked out of 6.00

We compare two different divide and conquer algorithms for the same problem.

Algorithm A: On an array of size n, the algorithm performs $\Theta(n^3)$ operations in the divide step, producing 4 subarrays, each of size $\frac{n}{16}$. The time taken to combine the output of these subproblems is $\Theta(n^2)$.

Algorithm B: On an array of size n, the algorithm performs $\Theta(n)$ operations in the divide step, to produce 2 subarrays, each of size $\frac{n}{2}$. The time taken to combine the output of the subproblems is $\Theta(n)$.

Algorithm C: On an array of size n, the algorithm produces 8 subarrays, each of size $\frac{n}{2}$ performing $\Theta(n^{2.5})$ operations for the divide step. The time taken to combine the output of the subproblems is $\Theta(n^2)$.

1) Write down the recurrences for the running time A(n) for algorithm A, B(n) for algorithm B, and C(n) for algorithm C on inputs of size n. You do not need to write out the base case.

2) Solve for A(n) using the master method to derive a Theta bound on worst case running time of algorithm A.

3) Solve for B(n) using the master method to derive a Theta bound on worst case running time of algorithm B.

4) Solve for C(n) using the master method to derive a Theta bound on worst case running time of algorithm C.

Master Method

The recurrence $T(n) = aT(\frac{n}{b}) + \Theta(n^c)$ with base case

$T(1) = \Theta(1)$ has the solution

Case 1 $\log_b a > c$ then $T(n) = \Theta(n^{\log_b(a)})$

Case 2 $\log_b a = c$ then $T(n) = \Theta(n^{\log_b(a)} \log(n))$

Case 3 $\log_b a < c$ then $T(n) = \Theta(n^c)$

Part 1

The recurrences for algorithms $A(n), B(n), C(n)$ are:

$$A(n) = 4T\left(\frac{n}{16}\right) + \Theta(n^3)$$
$$B(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$
$$C(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2)$$

For the rest of this problem, the quantity $\log_b(a)$ will be referred to as ϵ.

Part 2

Looking at the recurrence relation for $A(n)$, we can confidently say that for this algorithm; $a = 4, b = 16, c = 3$. When calculating the quantity ϵ, we find that $\epsilon = 0.5$ and this in turn is less than the quantity c. So, because of this we must adhere to **Case 3** and thus the solution using the Master Method is then:

$$A(n) = \Theta(n^3).$$

Part 3

Looking at the recurrence relation for $B(n)$, we can confidently say that for this algorithm; $a = 2, b = 2, c = 1$. When calculating the quantity ϵ, we find that $\epsilon = 1$ and this in turn is equal to the quantity c.

So, because of this we must adhere to **Case 2** and thus the solution using the Master Method is then:

$$B(n) = \Theta(n^{\log_2(2)} \log(n)) = \Theta(n \log(n)).$$

Part 4

Looking at the recurrence relation for $C(n)$, we can confidently say that for this algorithm; $a = 8, b = 2, c = 2$.

When calculating the quantity ϵ, we find that $\epsilon = 3$ and this in turn is greater than the quantity c. So, because of this we must adhere to **Case 1** and thus the solution using the Master Method is then:

$$C(n) = \Theta(n^{\log_2(8)}) = \Theta(n^3).$$

Question **20**

Complete

Marked out of 6.00

We are interested in performing quicksort where the pivoting strategy uses a median of 9 medians approach that works as follows.

Here is the outline of how the quicksort algorithm will proceed.

(A) Divide array of size n into n/9 parts of 9 elements each.

(B) Use insertion sort to sort each 9 element subarray and extract the median.

(C) Recursively apply median finding to find the median of the n/9 medians from part (B)

(D) Use the result from (C) as the pivot.

(E) Perform partitioning using the pivot.

(F) Recursively sort the two parts.

1. What is the running time in terms of the array size n for parts (A) and (B) of the algorithm proposed above? You can write an answer in text such as "Theta(n^2)" or in latex as $\Theta(n^2)$.

For part (A), the runtime of dividing an array in n / 9 parts is going to be $\Theta(n/9)$. The worst case runtime of running insertion sort on these subarrays is going to be $\Theta(n^2)$.

2. Assuming all array elements are unique, how many elements in the array can be guaranteed strictly smaller than the "median of medians" found in step (C)? How many array elements are guaranteed to be strictly larger?

The number of elements that can be guaranteed to be smaller than the medium is $\frac{5n}{18}$ and the number of elements that can be guaranteed to be larger than the medium is $\frac{5n}{18}$ as well.

3. If the pivot element in (C) is used to partition, what is the size of the larger partition in the worst case?

The larger partition in the worst case is going to be $\frac{13n}{18}$.

4. Let P(n) be the complexity of finding the pivot using the median of median trick. Write down a recurrence for P(n). This recurrence should only account for finding the Pivot.

The recurrence relation for finding the pivot using the median of medians trick is:

$$P(n) = P\left(\frac{13n}{18}\right)$$

5. What is the closed form for P(n)? Use Theta notation for your answer.

If we include the dividing the array up into 9 equal parts and then finding the pivot, the recurrence relation $P(n)$ is then known to be

$$P(n) = P\left(\frac{n}{9}\right) + P\left(\frac{13n}{18}\right).$$

6. Let S(n) be the worst case complexity for quicksort algorithm as a whole on an array of size n. Write down a recurrence for S(n) using the closed form answer for P(n) derived in step 5.

The worst case runtime of quicksort on an array of size n is known to be $\Theta(n^2)$. So, when combining this with the previous part we can then say the recurrence relation for $S(n)$ is then

$$S(n) = S\left(\frac{n}{9}\right) + S\left(\frac{13n}{18}\right) + \Theta(n^2).$$

Question **21**

Complete

Marked out of 5.00

We are writing a simple game AI for guiding Jane the dynamic programmer to dynamic programming greatness. Like before, our goal is to determine an optimal set of courses Jane can take. However, a few things have changed for Jane.

Jane's local university offers n different courses on dynamic programming which are numbered $1, \ldots, n$. Each course $1 \leq i \leq n$ has a fee f_i which Jane must pay to take the course. Likewise, each course $1 \leq i \leq n$ has a number of credits C_i which are rewarded upon completion.

Jane starts off having completed the very first course, course 1. Her goal is to complete the final course, course n, and maximize the total number of credits earned. She also doesn't want to go broke along the way.

Her university has strict policies about which courses a student can take: After completing a course, a student may only move forward exactly 1, 4, 5 or 11 courses. (E.g. After a student completes course 6, they may only take course 7, 10, 11 or 17 provided that such courses exist.)

We want to help compute the maximum amount of credits Jane can earn before completing course n.

Let maxCredits(i, B) denote the maximum amount of credits that Jane can earn after completing course n, if course i is the highest number course she has already completed and she wants to avoid going broke along the way (not spending more than B dollars).

The problem below has 6 parts (A)-(E).

(A) What is the value of maxCredits(i, B) if i = n and B >=0?

In this case we would set maxCredits(i, B) = **INFINITY** if B >= 0.

(B) What is the value of maxCredits(i, B) if B <0?

In this case we would set maxCredits(i, B) = **0** if B < 0.

(C) Write down a recurrence for maxCredits(i, B) . You may edit/fill in the following template:

maxCredits(i, B) = MAX(1 + maxCredits(i + 1, B - 1) , 4 + maxCredits(i + 4, B - 4) , 5 + maxCredits(i + 5, B - 5) , 11 + maxCredits(i + 11, B - 11))

(D) We wish to design a memo table for memoizing the recurrence above. What data structure (besides a hashtable) can we use to memoize maxCredits?

In this case, we need to essentially have a two dimensional memo table. Because of this, the best choice of a data structure for our memo table would be a **two dimensional array**.

(E) Write down a pseudocode for filling up this memo table.

```
Iterate over the possible classes:
    Iterate over the dollar amounts:
        Calculate maxCredits(i, B)
        Store result for the ith column and jth row in memo table
Repeat process until all possible values have been examined
```

Question **22**

Complete

Marked out of 4.00

This question asks you to prove that the problem of factoring a number n belongs to NP.

Here is the decision version of the factoring problem.

Input: Number n in binary, and index k

Output: Yes, if the k^{th} bit of the smallest prime factor of n is 1, NO, otherwise.

Prove that the problem is in the class NP.

(Note: we are not asking you to show that it is NP-complete. That is an open problem to date whose solution will make you really famous).

Here we are seeking to show that factoring a number is non-deterministic polynomial (NP). To show that this problem is indeed NP, we must show that factoring a number can be done in polynomial time.

Take for example the number 10, to determine the factors for 10 we must start from 1 and go up to 10. Along the way, we must only keep values where there is no remainder in the division step starting from 1. This step will at most only taking 10 operations.

From the array of values that were computed in the previous step, we can simply find the minimum value of these values, return YES if the smallest value is 1 and NO otherwise. Finding the minimum value of a set of numbers is known to be done in polynomial time.

So, extrapolating this to larger numbers, we are only going to perform n divisions of that number. Then, calculating the minimum of these numbers is know to be possible in polynomial time. Returning YES or NO can be done in constant time once the smallest value has been found from the list.

Thus, we can confidently say that **this problem is NP**.

Question **23**

Complete

Marked out of 4.00

The knapsack problem is given by the following inputs:

(A) Set of item weights $w_1, \ldots, w_n$
(B) Set of item values $v_1, \ldots, v_n$
(C) Weight limit W
(D) Value limit V.

The Knapsack problem asks you if it is possible (YES/NO) to select a subset of items 1 to n, such that the total weight of the selected items is less than or equal to W and the total value of the selected items is at least V.

The single inequality 0-1 ILP problem with a single inequality is given by the following inputs:

(A) 0-1 ILP Problem with single inequality (maximize an objective function subject to inequality and the variables are all either 0 or 1)

$$\begin{array}{llll} max & c_1x_1 + \cdots c_nx_n & & \leftarrow \text{Objective Function} \\ s.t. & a_{11}x_1 + \cdots + a_{1n}x_n & \leq b_1 & \leftarrow \text{Inequality Constraint} \\ & x_1, \ldots, x_n & \in \{0,1\}^n & \leftarrow \text{All variables are 0 or 1.} \end{array}$$

(B) Objective Limit K

The ILP Output:

YES if there exists 0-1 values for $x_1, \ldots, x_n \in \{0,1\}^n$ such that the inequality is satisfied and the value of the objective function is greater than or equal to K.
NO otherwise

-----

1. Suppose you are given that Knapsack is NP-Complete, show that 0-1 ILP is NP-Complete.

To show that the 0-1 ILP is NP-Complete, we can draw comparisons from the Knapsack Problem since it is known that the Knapsack Problem is NP-Complete. The strategy will be to reduce the Knapsack Problem down to where it can be utilized in the context of the 0-1 ILP problem. Doing so will show that since we reduced a known NP-Complete problem down and used it in a different context, this new problem must also be NP-Complete.

In the Knapsack problem, we are dealing with items from 1 to n and returning a YES or NO if the total weight of the items is less than W and greater than or equal to V. In the 0-1 ILP problem, the inequality constraint is equivalent to the weight W in the Knapsack problem and the objective limit is equivalent to the value V in the Knapsack problem. The values that exist in the list of x in the 0-1 ILP problem is equivalent to the items that are present in the Knapsack problem.

If we select the values from the objective function, check that there exists 0-1 values in the list such that the inequality constraint is satisfied, and then compare that the value of the objective function is greater than or equal to K, we are essentially doing the same procedure that is used in the Knapsack problem but in a different context.

Since we are using the same procedure as the Knapsack problem in the context of the 0-1 ILP problem, and we know that the Knapsack problem is NP-Complete, we can then conclude that the 0-1 ILP problem is **NP Complete**.