Quiz#5-DynamicProgg&ApproxAlgo-12thDec2022

p22cs013@coed.svnit.ac.in Switch account

Your email will be recorded when you submit this form

Quiz#5-DynamicProgg&ApproxAlgo-12th Dec2022

Quiz#5-DynamicProgg&ApproxAlgo-12thDec2022
Scheduled for 12th Dec, 06:00 pm
1. The quiz must be attempted using your SVNIT email ID only. If attempted using any other email ID, it would NOT be considered. There will not be any exceptions to this.
2. Please attend the quiz that is assigned to you.
3. **Total Questions: 30, Total Marks: 60. TimeDuration: 40 minutes.**
4. Any quiz that is received after 06:42 pm, shall NOT be graded and shall be considered as Not Attempted. Therefore, do not continue attending till 06:41 pm - stop at 06:40 pmand let the quiz be submitted and received in the next two minutes.
5. At times, the latex notations are used in writing a question - interpret them as in case of latex e.g. x simply means an identifier x - $ coming in from latex math mode. Similarly α means the greek letter alpha.

Approximation algorithms provide bounds on the quality of the solution mathematically. This statement is ______. 2 points

- False
- none of these
- Can't say
- True

Whether the maximization optimization problem A(I) or minimization optimization problem A(I) on instance I, the approximation ratio ρ, closer to the value one, ________ is the approximation algorithm A(I)

2 points

- ○ better
- ○ equal to the OPT
- ○ does not matter
- ○ poorer

Suppose there is a row of n coins whose values are some positive integers c_1, c_2 ,...,cn, not necessarily distinct. Let the goal be to pick up the maximum amount of money subject to the constraint that no two coins adjacent in the initial row can be picked up. Then the recurrence used by the dynamic programming algorithm for the problem is ___________

2 points

- ○ F[n]←max(c[n-2] + F[n−1], F[n−1])
- ○ F[n]←max(c[n-1] + F[n−2], F[n−1])
- ○ F[n-1]←max(c[n-1] + F[n−2], F[n−1])
- ○ F[n]←max(c[n] + F[n−2], F[n−1])

Consider the 0/1 Knapsack problem, having items with weights w_1,...,w_i , values v_1,...,v_i and knapsack capacity W. Consider an instance defined by the first i items, 1 <= i <= n, with weights w_1 ,...,w_i , values v_1 ,...,v_i , and knapsack capacity j, 1 <= j <= W. Let F(i, j) be the value of an optimal solution. Then, the dynamic programming recurrence expression for computing the maximum profit out of subset of items 1...i with weight limit j would be given by _____________.

2 points

- ○ F(i,j)=max{F(i−1,j), vi + F(i−1,j−wi)} if j−wi ≥0, F(i,j)=F(i,j), if j−wi < 0
- ○ F(i,j)=max{F(i−1,j−wi), vi + F(i−1,j)} if j−wi ≥0, F(i,j)=F(i−1,j−wi), if j−wi < 0
- ○ F(i,j)=max{F(i,j), vi + F(i,j−wi)} if j−wi ≥0, F(i,j)=F(i−1,j), if j−wi < 0
- ○ F(i,j)= max{F(i−1,j), vi + F(i−1,j−wi)} if j−wi ≥0, F(i,j)=F(i−1,j), if j−wi < 0

You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \cdots <$ an, where each ai is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance an), which is your destination. You'd ideally like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel **x** miles during a day, the *penalty* for that day is **(200 − x)^2**. You want to plan your trip so as to minimize the total penalty—that is, the sum, over all travel days, of the daily penalties. Then, an efficient algorithm that determines the optimal sequence of hotels at which to stop would use the recurrence viz. ________________. [With OPT(i), denoting be the minimum total penalty to get to hotel i. hotel i.]

2 points

- ○ OPT(i) = min { OPT(j) + (200 − (a_j − a_i))^2 } [0 ≤ j < i]
- ○ OPT(i) = min { OPT(i-1) + (200 − (a_j − a_i))^2 } [0 ≤ j < i]
- ○ OPT(i) = min { OPT(j) + (200 − (a_j − a_i))^2 } [0 >= j > i]
- ○ OPT(j) = min { OPT(i) + (200 − (a_j − a_i))^2 } [0 >= j > i]

Consider an airline reservation application. An approximation algorithm for selling the tickets used therein, in an instance, sells 10 airplane tickets with a profit of 10. However, the optimum profit that could have been made on selling 10 air tickets is 20. Then, the approximation ratio ρ here is ________. 2 points

- 1
- 2
- 4
- 0.5

Designing an algorithm using Dynamic programming involves ________. 2 points

- breaking up a problem into a series of overlapping sub-problems, and build up solutions to larger and larger sub-problems, that can be used to solve many problems in time O(n^2) or O(n^3) for which a naive approach would take exponential time.
- breaking up a problem into a series of overlapping sub-problems, and build up solutions to larger and larger sub-problems, that can be used to solve many problems that otherwise require time O(2^n) or higher.
- using a top up design approach to solve the original problem.
- breaking up a problem into a series of smaller sub-problems, to solve the smaller subproblems and use the solution in conquering the original problem, that can be used to solve many problems in time O(n^2) or O(n^3) for which a naive approach would take exponential time.

Suppose the job of a firm is to manage the construction of billboards on the Surat-Dumas Gauravpath that runs east-west for **M** kms. The possible sites for the billboards are given by numbers **x1, x2, x3, x4, x5,…. xn** each in the interval **[0…M]. xi's** are indicating the position of the billboards along the Gauravpath in kms, measured from its western end. If it is decided to place a billboard at location ***xi***, then the firm receives a revenue of **ri>0**. Regulations of the SMC required that no two of the billboards be within less than or equal to t kms of each other. Thus, as part of the optimization problem, the firm has to decide where to place the billboards at a subset of the sites **x1, x2, x3, x4, x5,…. xn** so as to maximize the total revenue, subject to this constraint. Then, given the Input : M = 15, separation distance t=2 kms, Distances x[i] = {6, 9, 12, 14}, revenue[i] = {5, 6, 3, 7}. The Dynamic Programming algorithm shall place the billboards at a distances viz. _________ to give maximum revenue of _______.

2 points

- 6 & 9 & 14, 18
- 6 & 9 & 12, 14
- 9 & 12 & 14, 16
- none of these

Let, OPT(I) denote the value of an optimal solution to the problem under consideration for input I and let OPT(G) denote the length of a shortest tour on a point set G in the Travelling Salesperson Problem. Then, when solved using the Approximation algorithm design, the value of OPT_approx(G) is _______ OPT(G).

2 points

- equal to
- could be lesser than or greater than
- lesser than
- greater than

Consider execution of a project that requires a certain number of skills (set X). Each team member possesses a subset of the skills. If the optimal value of the team members is 10 whereas the approximation algorithm selects the value 15. Then, the approximation ratio is __________.

2 points

- 15
- 0.66
- 1.5
- 6

The direct ______ approach to finding a solution to such a recurrence leads to an algorithm that solves common subproblems more than once and ______. The classic dynamic programming approach, on the other hand, works ______: it fills a table with solutions to all smaller subproblems, but each of them is solved only once. This limitation of the bottom up approach can be resolved using ____________.

2 points

- top-down, hence is very inefficient, top-down, memoization
- bottom-up, hence is very efficient, top-down, memoization
- top-down, hence is very efficient, top-down, memory functions
- top-down, hence is very inefficient, bottom up, memory functionstion 1

Given a weighted graph with n nodes, finding the shortest path that visits every node exactly once (i.e. Traveling Salesman Problem) that takes \fillin time using the brute-force approach, would require the time \fillin using the dynamic programming solution shown here.

2 points

- $O(n!)$, $O(n^2 * 2^n)$
- $O(2^n)$, $O(n!)$
- $O(n!)$, $O(n^2)$
- $O(2^n)$, $O(n^2)$

Shambu wants to make money in the share market. To limit his damages, he has decided to focus on a single company. He will always buy and sell 100 shares at a time, and he will never hold more than 100 shares. Before entering the share market, he did some research on the share price over the last few days to determine the maximum profit he could have made. Suppose the price of 100 shares over a period of 10 days varied as shown in the figure, and he holds no shares at the start of day 1. A greedy strategy is to buy when the price is low and sell when the price is high. The maximum money he can make in this case is 13: buy on day 2, sell on day 3 (+3), buy on day 4, sell on day 6 (+5), buy on day 7, sell on day 8 (+5). To discourage speculation, the stock market has decided to charge a transaction fee. Each time 100 shares are bought or sold, Rs 1 has to be paid to the stock market. Here is how we recursively compute Shambu's best strategy now over a period of N days. (1) On day i, Shambu can either buy or not buy 100 shares.(2) If he does not buy on day i, it is as though he started his transactions directly on day i+1. (3) If Shambu does buy on day i, the money he makes depends on when he sells the shares on days i+1,...,N. Let Profit[i] be the money Shambu can make on days i,i+1,...,N, assuming he has no shares at the start of day i. Then, ________ of the following is correct to describe this strategy.

2 points

Day	1	2	3	4	5	6	7	8	9	10
Price	11	7	10	9	13	14	10	15	12	10

- max(Profit[i+1], Profit[i+2] + Price[i+1] - Price[i] - 2, Profit[i+3] + Price[i+2] - Price[i] - 2, ... Profit[N] + Price[N-1] - Price[i] - 2, Price[N] - Price[i] - 2)
- max(Profit[i+1], Profit[i+2] + Price[i+1] - Price[i] - 2, Profit[i+3] + Price[i+2] - Price[i] - 2, ... Profit[N] + Price[N-1] - Price[i] - 2)
- max(Profit[i+2] + Price[i+1] - Price[i] - 2, Profit[i+3] + Price[i+2] - Price[i] - 2, ... Profit[N] + Price[N-1] - Price[i] - 2, Price[N] - Price[i] - 2)

The 0/1 Knapsack problem - given a specific combinations of the items and defined in terms of the values of the input, can be solved in time \fillin using the dynamic programming. [n= number of items, W=knapsack carrying capacity

2 points

- $O(n^2)$
- $O(n)$
- $O(n\;lg\;n)$
- O(nW)

Consider the IndependentSet-Approximation algorithm. Given a graph G=(V,E) with an optimal independent set of size 15, the IndependentSet-Approximation algorithm could give the answer as ___________, yielding the approximation ratio ρ as ______.

2 points

- 8, 1.875
- 30, 2
- 8, 0.53
- 30, 0.5

Consider the Clique-Approximation algorithm. Given a graph G=(V,E) with a maximum clique of size 10, the Clique-Approximation algorithm could give the answer as _______, yielding the approximation ratio ρ as _______.

2 points

- 8, 1.25
- 12, 1.2
- 8, 0.8
- 12, 0.833

________ is a good order to compute the Profit[i] using the dynamic programming, in a typical problem. 2 points

- From Profit[1] to Profit[N]
- Either Profit[1] to Profit[N] OR Profit[N] to Profit[1]
- None of these
- From Profit[N] to Profit[1]

There are three main directions to solve NP-hard discrete optimization problems viz. Integer programming techniques and _______ and __________. 2 points

- Heuristics, Probabilistic algorithms
- Approximation algorithms, Backtracking approach
- Heuristics, Branch and Bound
- Heuristics, Approximation algorithms

Consider an instance of the knapsack problem with item values $v_1, v_2, \ldots, vn$, item sizes $s_1, s_2, \ldots, sn$, and knapsack capacity C, and an optimal solution $S \subseteq \{1, 2, \ldots, n\}$ with total value $V = \Sigma i \varepsilon S(vi)$. Then the statements from the following that hold for the set (S -{n}) i.e the optimal solution that does not make use of the last item n, are ____________________ 2 points

a) It is an optimal solution to the subproblem consisting of the first (n-1) items and knapsack capacity C.

b) It is an optimal solution to the subproblem consisting of the first (n - 1) items and knapsack capacity (C - vn). c) It is an optimal solution to the subproblem consisting of the first (n - 1) items and knapsack capacity (C - sn). d) It might not be feasible if the knapsack capacity is only (C - sn).

- c
- b
- a
- d

Suppose the job of a firm is to manage the construction of billboards on the Surat-Dumas Gauravpath that runs east-west for **M** kms. The possible sites for the billboards are given by numbers **x1, x2, x3, x4, x5,…. xn** each in the interval **[0…M]. xi's** are indicating the position of the billboards along the Gauravpath in kms, measured from its western end. If it is decided to place a billboard at location ***xi***, then the firm receives a revenue of **ri>0**. Regulations of the SMC required that no two of the billboards be within less than or equal to t kms of each other. Thus, as part of the optimization problem, the firm has to decide where to place the billboards at a subset of the sites **x1, x2, x3, x4, x5,…. xn** so as to maximize the total revenue, subject to this constraint. Then, given the Input : M = 20, separation distance t=5 kms, Distances x[i] = {6, 7, 12, 13, 14}
revenue[i] = {5, 6, 5, 3, 1}. The Dynamic Programming algorithm shall place the billboards at a distance _________ and _________ to give maximum revenue of _______.

2 points

- 6,14, 6
- 12, 13, 8
- 7,13, 9
- 6,12, 10

Suppose you own two stores, A and B. On each day you can be either at A or B. If you are currently at store A (or B) then moving to store B the next day (or A) will cost C amount of money. For each day i, i = 1, . . . , n, we are also given the profits PA(i) and PB(i) that you will make if you are store A or B on day i respectively. Give a schedule which tells where you should be on each day so that the overall money earned (profit minus the cost of moving between the stores) is maximized. Then, the recurrence used by the Dynamic Programming algorithm is as follows _____________. [Assume TA[i] gives the most profitable schedule for days i, . . . , n]

2 points

- TA[i] = PB(i) + max(TB[i + 1], TB[i + 1] − C).
- TA[i] = PA(i) + max(TB[i + 1], TA[i + 1] − C).
- TA[i] = PA (i) + max(TA[i + 1], TB[i + 1] − C).
- TA[i] = PB(i) + max(TB[i + 1], TA[i + 1] − C).

Consider two approximation algorithms viz. A and B for the Set Cover problem. The optimal set cover for a given set S is of the size 10, whereas the algorithm A and B output the size of the set cover as 12 and 16, respectively. Then, you would choose the algorithm ________ because its approximation ratio ρ is _______ as compared to that of _________.

2 points

- B, greater, A
- A, lesser, B
- B, lesser, A
- A, greater, B

In Hochbaum's words: Trading-off ________ in favour of ______ is the paradigm of approximation algorithms

2 points

- optimality, tractability
- efficiency, optimality
- tractability, efficiency
- tractability, optimality,

Suppose there is a row of n coins whose values are some positive integers c_1, c_2 ,...,cn, not necessarily distinct. Let the goal be to pick up the maximum amount of money subject to the constraint that no two coins adjacent in the initial row can be picked up. Then given a coin row as 5, 1, 2, 10, 6, 2, the maximum amount that can be picked up is ___________

2 points

- 15
- 5
- 18
- 7

Consider the following expression wherein, 1 ≤ Max (Ax(I)/OPT(I), OPT(I)/Ay(I)) ≤ ρ(n), where OPT(I) is the optimal algorithm to solve a problem on instance I, whereas Ax and Ay are two different approximation algorithms to solve two different problems. This relation is _______ and Ax is solving a __________ problem whereas Ay is solving a _________ problem.

2 points

- false, maximization, minimization
- true, minimization, maximization
- true, maximization, minimization
- false, minimization, maximization

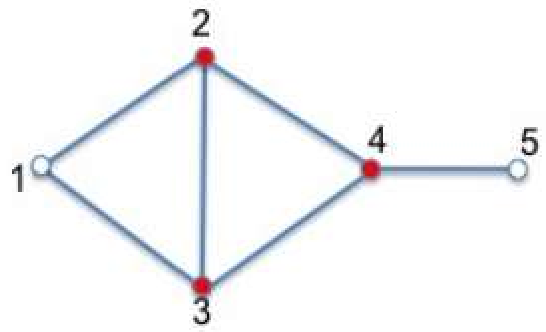
The complexity of the recursive algorithm implementing the recursive Fibonacci function is _____.

2 points

- O(1.61803^n)
- O(n^2)
- O(2n)
- O(n)

Consider the graph shown in the figure. The optimal vertex cover of this graph is _______ consisting of _______ vertices.

2 points



- 3, {2,3,5}
- 4,{1,2,3,4}
- 3, {1,2,3}
- 2,{1,2}

A recursive algorithm that is based on the recurrence expression for the following problem can be given as shown in the code snippet, as shown in the figure. Problem description: Consider the variation of the Interval scheduling problem studied in the chapter on Greedy design, as follows: We have n requests labeled 1, . . . , n, with each request i specifying a start time s_i and a finish time f_i. Each interval i now also has a value, or weight v_i . Two intervals are compatible if they do not overlap. The goal of our current problem is to select a subset S ⊆ {1, . . . , n} of mutually compatible intervals, so as to maximize the sum of the values of the selected intervals, sigma{i ϵ S} v_i.

2 points

```
Compute-Opt(j)
If j = 0 then
        Return 0
Else
        Return max(v_j + Compute - Opt(p(j)), Compute - Opt(j1))
Endif
```

- it really solves n^2 different subproblems, polynomial time
- it really solves $n + 1$ different subproblems, exponential time
- it really solves $n + 1$ different subproblems, polynomial time
- it really solves $n!$ different subprobelms, exponential time

Consider the instance of the coin-changing problem with the given specifications as shown in the figure here. The recurrence that is used to populate the given table is __________. The number of coins required for the making up an amount of 8 paisa is _________.

2 points

	Amount	0	1	2	3	4	5	6	7	8
0	$d_0=0$	0	0	0	0	0	0	0	0	0
1	$d_1=1$	0	1	2	3	4	5	6	7	8
2	$d_2=4$	0	1	2	3	1	2	3	4	2
3	$d_3=6$	0	1	2	3	1	2	1	2	2

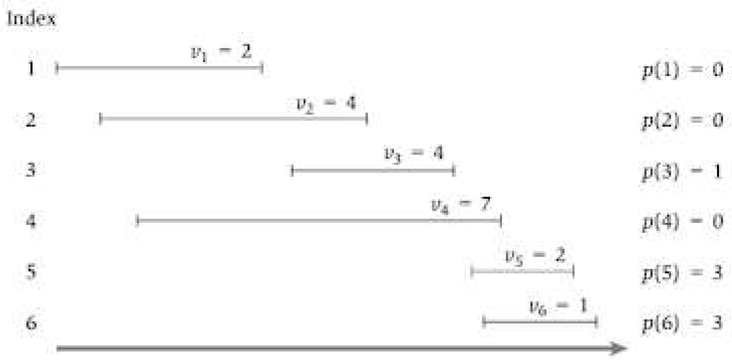
- $c[i, j] = min {1+ c[i, j-d_i], c[i-1, j]}$, 2
- $c[i, j] = min {1+ c[i, j], c[i, j]}$, 3
- $c[i, j] = min {1+ c[i, j], c[i, j]}$, 2
- $c[i, j] = min {1+ c[i, j-d_i], c[i-1, j]}$, 3

In the ________ approach, to reduce the time required for solving a problem, we relax the problem, and obtain a feasible solution “close” to an optimal solution i.e. we compromise on optimality for a good feasible solution. 2 points

- none of these
- heuristics-based design
- randomized algorithms design based
- approximation algorithms design

Consider the Interval scheduling problem studied in the chapter on Greedy design. In a variation of the same problem, consider the following: We have n requests labeled 1, . . . , n, with each request i specifying a start time s_i and a finish time f_i. Each interval i now also has a value, or weight v_i . Two intervals are compatible if they do not overlap. The goal of our current problem is to select a subset S ⊆ {1, . . . , n} of mutually compatible intervals, so as to maximize the sum of the values of the selected intervals, sigma{i ϵ S} v_i. This problem ______ be solved using the greedy design technique by _____. When attempting to solve using the dynamic programming, the recurrence on which an initial solution could be based could be described as ______. [Assume that v_j is the value of the jth interval, OPT(j) is the value of the optimal solution in which there are {1....j} input intervals]

2 points



- cannot, not applicable, OPT(j) = max(p(j) + OPT(p(j)), OPT(j−1)).
- can, sorting intervals in order of their finish time, OPT(j) = max(v_j + OPT(p(j)), OPT(j−1)).
- cannot, not applicable, OPT(j) = max(v_j + OPT(p(j)), OPT(j−1)).
- can, sorting intervals in order of their finish time, OPT(j) = max(p(j) + OPT(v(j)), OPT(j−1)).

Page 2 of 2

Back Submit Clear form