(9) 50

Difference b/w Dynamic Programming and Greedy algo :→

The choice made by a greedy algo. may depend on choices made so far but not on future choices, ~~or all the solutio~~ It iteratively makes one greedy choice after another, reducing each given problem into a smaller one. greedy algo. never reconsiders its choices.

Dynamic programming is exhaustive and is guaranteed to find the solution. After every stage, dynamic programming makes decision based on all the decisions made in the previous stage and may reconsider the previous stage's algorithmic path to solution.

eg. let's say that you have to get from pt. A to pt. B as fast as possible, in a given city, during rush hour. A dynamic programming algo. will look into the entire traffic report, looking into all possible combinations of roads you might take, and will only then tell which way is the fastest. of course, you might have to wait for a while until the algo. finishes and only then you can start driving. The path you will take be the fastest one.

On the other hand, a greedy algo. will start you driving immediately and will pick the road that looks the fastest at every intersection.

As you can imagine, this strategy might not lead to the fastest arrival time, since you might take some easy streets and then find yourself hopelessly stuck in a traffic jam.

Prim's algo and Krushal's algo for minimum spanning tree are greedy algo's.

Dijkstra Algo to find shortest path is greedy algo.
Bellman Ford " " " " is Dynamic prog.

Dijkstra algo greedily selects the min. weight node that has not yet been processed and performs this relaxation process on all of its outgoing edges. But Bellman ford algo. relaxes all the edges and does this |V|-1 times where |V| is the no. of vertices in the graph.

Bellman ford algo is used in distance Vector routing algo. Bellman ford algo can also have -ve weight (No -ve cycle reachable from source S) but Dijkstra algo cannot have -ve weights.

(Floyd Warshall → D.P.) calculates shortest distance b/w nodes while Bellman Ford algo. calculates shortest path distance from source node to other vertices.

In Floyd Warshall any node can be a source/destination
In Bellman Ford only one node can be a source/destination. → Dynamic Programming (F.W, B.F)

Dynamic Routing Protocols

Distance Vector | Link State

<u>Distance Vector Routing Protocol</u> :→ Distance Vector Routing Protocols base their decisions on the best path to a given destination based on the distance. The route with the least no. of hops to a given n/w is concluded to be the best route towards that n/w. The vector shows the direction to that specific n/w. Distance vector protocols send their entire routing table to directly connected neighbors. examples:→ RIP (Routing Information Protocol) and IGRP (Interior Gateway Routing Protocol)

<u>Link State Routing Protocols</u> :→ Link state protocols are also called shortest path first protocols. Link state routing protocols have a complete picture of the n/w topology. Hence they know more about the whole n/w than any distance vector protocol. Link state protocols send info. about directly connected links to all the routers in the n/w. e.g. OSPF – Open Shortest path first. This protocol uses Dijkstra algo. to build a routing table.

Dec 14(III) ~~12~~ 53

(35) Consider the problem of a chain of three matrices. Suppose that the dimensions of matrices are 10 X 100, 100 X 5 and 5 X 50 respectively. There are two diff. ways (i) $((A_1 A_2) A_3)$ (ii) $(A_1 (A_2 A_3))$

Computing the product according to first is <u>10</u> times faster in comparison to the second.

Sol: $$((A_1 A_2) A_3) = \overset{A_1 A_2}{10 \times 100 \times 5} + \overset{A_1(A_2 A_3)}{10 \times 5 \times 50} = 7500$$

$$(A_1 (A_2 A_3)) = \overset{A_2 A_3}{100 \times 5 \times 50} + \overset{A_1(A_2 A_3)}{10 \times 100 \times 50} = 75000$$

(36) Suppose that we have numbers b/w 1 and 1000 in a binary search tree and we want to search for the number 365. Which of the following sequences could not be the seq of nodes examined.

(A) 4, 254, 403, 400, 332, 346, 399, 365

(B) 926, 222, 913, 246, 900, 260, 364, 365

(C) 927, 204, 913, 242, 914, 247, 365

(D) 4, 401, 389, 221, 268, 384, 383, 280, 365

927

204

913

242

914 → wrong as it is on the left of 913 ∴ it should be less than 913

* Path from root to 365 is given.

Dec 10 (54)

(21) What is the maximum no. of nodes in a B-tree of order 10 of depth 3 (root at depth 0)?

Ans $10^0 + 10^1 + 10^2 + 10^3 = 1111$

(22) A strictly binary tree with 27 nodes has 14 null branches. [n leaves has $2n-1$ nodes, so $2n-1 = 27$, $n = 14$]

(2) An undirected graph possesses an ~~eleulo~~ eulerian circuit if and only if it is connected and its vertices are.

(a) all of even degree (b) all of odd degree

(c) of any degree (d) even in number.

(25) Which of the following can be the sequence of nodes examined in binary search tree while searching for key 88.

(a) 90, 40, 65, 50, 88

(b) 90, 110, 80, 85, 88

(c) 190, 60, 90, 85, 88

(d) 65, 140, 80, 70, 88

Dec 10

? (25) In a B tree of order 5, the following keys are inserted as follows:

7, 8, 1, 4, 13, 20, 2, 6 & 5

How many elements are present in the root of the tree Ans (2)

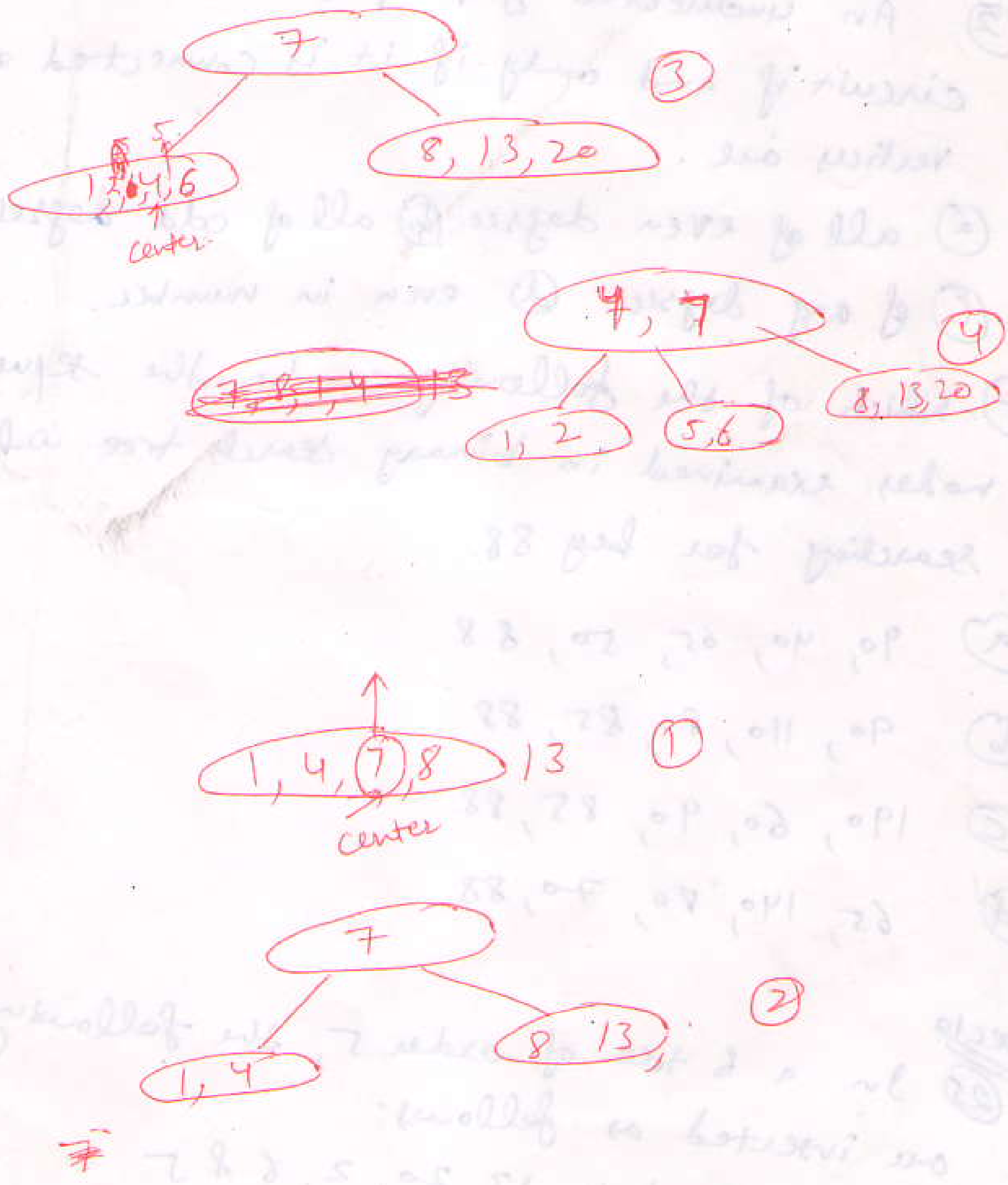
Dec14(III) ~~14~~ 55

(34) Dijkstra algo, which solves the single source shortest paths problem is a ______ and the floyd warshall algo which finds shortest paths b/w all pair of vertices is a .

Ans Greedy algo, Dynamic Programming algo.

Dec10

(25)



56

Satisfiability of boolean formula

3-Conjunctive Normal form satisfiability

NP P NPC

$P \cap NPC = \phi$

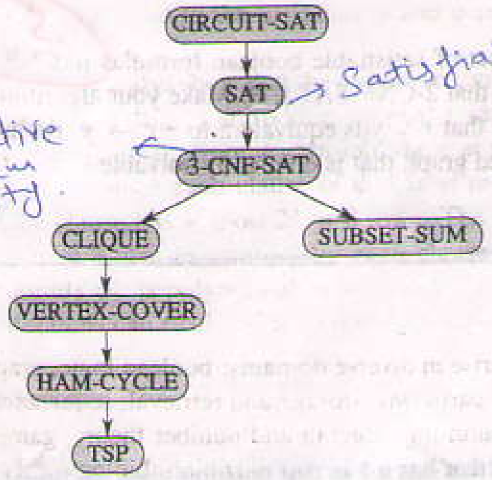


Figure 34.13 The structure of NP-completeness proofs in Sections 34.4 and 34.5. All proofs ultimately follow by reduction from the NP-completeness of CIRCUIT-SAT.

Proof To show that CLIQUE $\in$ NP, for a given graph $G = (V, E)$, we use the set $V' \subseteq V$ of vertices in the clique as a certificate for G. Checking whether V' is a clique can be accomplished in polynomial time by checking whether, for each pair $u, v \in V'$, the edge (u, v) belongs to E.

We next prove that 3-CNF-SAT $\leq_P$ CLIQUE, which shows that the clique problem is NP-hard. That we should be able to prove this result is somewhat surprising, since on the surface logical formulas seem to have little to do with graphs.

The reduction algorithm begins with an instance of 3-CNF-SAT. Let $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ be a boolean formula in 3-CNF with k clauses. For $r = 1, 2, \ldots, k$, each clause C_r has exactly three distinct literals l_1^r, l_2^r, and l_3^r. We shall construct a graph G such that ϕ is satisfiable if and only if G has a clique of size k.

The graph $G = (V, E)$ is constructed as follows. For each clause $C_r = (l_1^r \vee l_2^r \vee l_3^r)$ in ϕ, we place a triple of vertices v_1^r, v_2^r, and v_3^r into V. We put an edge between two vertices v_i^r and v_j^s if both of the following hold:

- v_i^r and v_j^s are in different triples, that is, $r \neq s$, and
- their corresponding literals are ***consistent***, that is, l_i^r is not the negation of l_j^s.

This graph can easily be computed from ϕ in polynomial time. As an example of this construction, if we have

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) ,$$

then G is the graph shown in Figure 34.14.

CLIQUE can be reduced from 3-CNF-SAT which is NP-complete problem. It means CLIQUE is NP hard

Now (1) CLIQUE can be verified in polynomial time

(2) CLIQUE is NP hard.

CLIQUE is NP complete.

57

satisfied. If S' includes a v_i' that has a 1 in that position, then the literal $\neg x_i$ appears in C_j. Since we have set $x_i = 0$ when $v_i' \in S'$, clause C_j is again satisfied. Thus, all clauses of ϕ are satisfied, which completes the proof. ■

Exercises

34.5-1

The ***subgraph-isomorphism problem*** takes two graphs G_1 and G_2 and asks whether G_1 is isomorphic to a subgraph of G_2. Show that the subgraph-isomorphism problem is NP-complete.

34.5-2

Given an integer $m \times n$ matrix A and an integer m-vector b, the ***0-1 integer-programming problem*** asks whether there is an integer n-vector x with elements in the set $\{0, 1\}$ such that $Ax \leq b$. Prove that 0-1 integer programming is NP-complete. (*Hint:* Reduce from 3-CNF-SAT.)

34.5-3

The ***integer linear-programming problem*** is like the 0-1 integer-programming problem given in Exercise 34.5-2, except that the values of the vector x may be any integers rather than just 0 or 1. Assuming that the 0-1 integer-programming problem is NP-hard, show that the integer linear-programming problem is NP-complete.

34.5-4

Show that the subset-sum problem is solvable in polynomial time if the target value t is expressed in unary.

34.5-5

The ***set-partition problem*** takes as input a set S of numbers. The question is whether the numbers can be partitioned into two sets A and $\bar{A} = S - A$ such that $\sum_{x \in A} x = \sum_{x \in \bar{A}} x$. Show that the set-partition problem is NP-complete.

34.5-6

Show that the hamiltonian-path problem is NP-complete.

34.5-7

The ***longest-simple-cycle problem*** is the problem of determining a simple cycle (no repeated vertices) of maximum length in a graph. Show that this problem is NP-complete.

3-COLOR is NP Complete

half 3-CNF satisfiability is NP Complete

(51) Suppose there are log n sorted lists of n/log n elements each. The time complexity of producing a sorted list of all these elements is (use heap data structure) 58

(a) $O(n \log\log n)$. (b) $\Theta(n \log n)$ (c) $\Omega(n \log n)$ (d) $\Omega(n^{3/2})$

Sol:→ Basically, we have to merge log n sorted lists of size n/log n each.

Using Minheap we can merge k sorted lists of size n each in

$$O(nk \log k)$$

Here $n = n/\log n$ $k = \log n$

$$O\left(\frac{n}{\cancel{\log n}} \times \cancel{\log n} \log\log n\right) = O(n \log\log n)$$

Note:→ In Minheap, the root is always less than or equal to children.