Recitation: Week 16

EE4033 Algorithms, Fall 2019

Instructor: Yao-Wen Chang, James Chien-Mo Li, and Iris Hui-Ru Jiang

2019-12-25

Presenter: 徐晨皓 Chen-Hao Hsu
r07943107@ntu.edu.tw

Instructions

- Deadline: 6:00 pm, January 3, 2020 (Fri.) @ BL-406

- **No late submission will be accepted!**

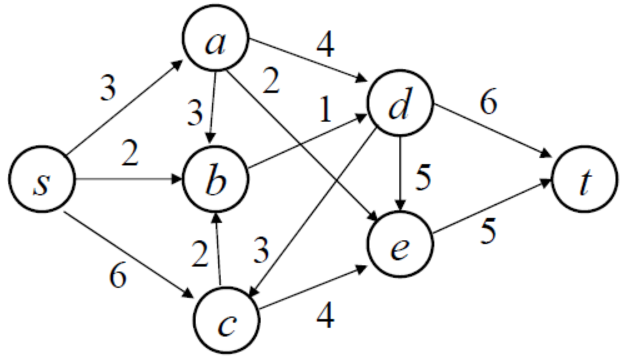
- Collaboration policy

  You can discuss the problems with other students, but you must write the final answers by yourself. Please specify all of your collaborators (names and student ID's) for each problem. If you solve some problems by yourself, please also specify "no collaborators".

  Problems without collaborator specification will not be graded!

1. The Edmonds-Karp Algorithm

1. In the flow network shown below, the number beside an edge denotes its corresponding capacity. Apply the **Edmonds-Karp** algorithm to find a maximum flow from s to t in the network. Show **every augmentation path** (but you **do NOT need to show the whole network** to save time) and **explain why the flow you found is maximum.**



- Use the Edmonds-Karp algorithm
- Show **every** augmenting path step by step
  - Optional: show the **residual networks** step by step
  - Optional: follow lexicographical order for BFS
- Maximum flow = ?
- Explain why the flow you found is maximum (Hint: the max-flow min-cut theorem)

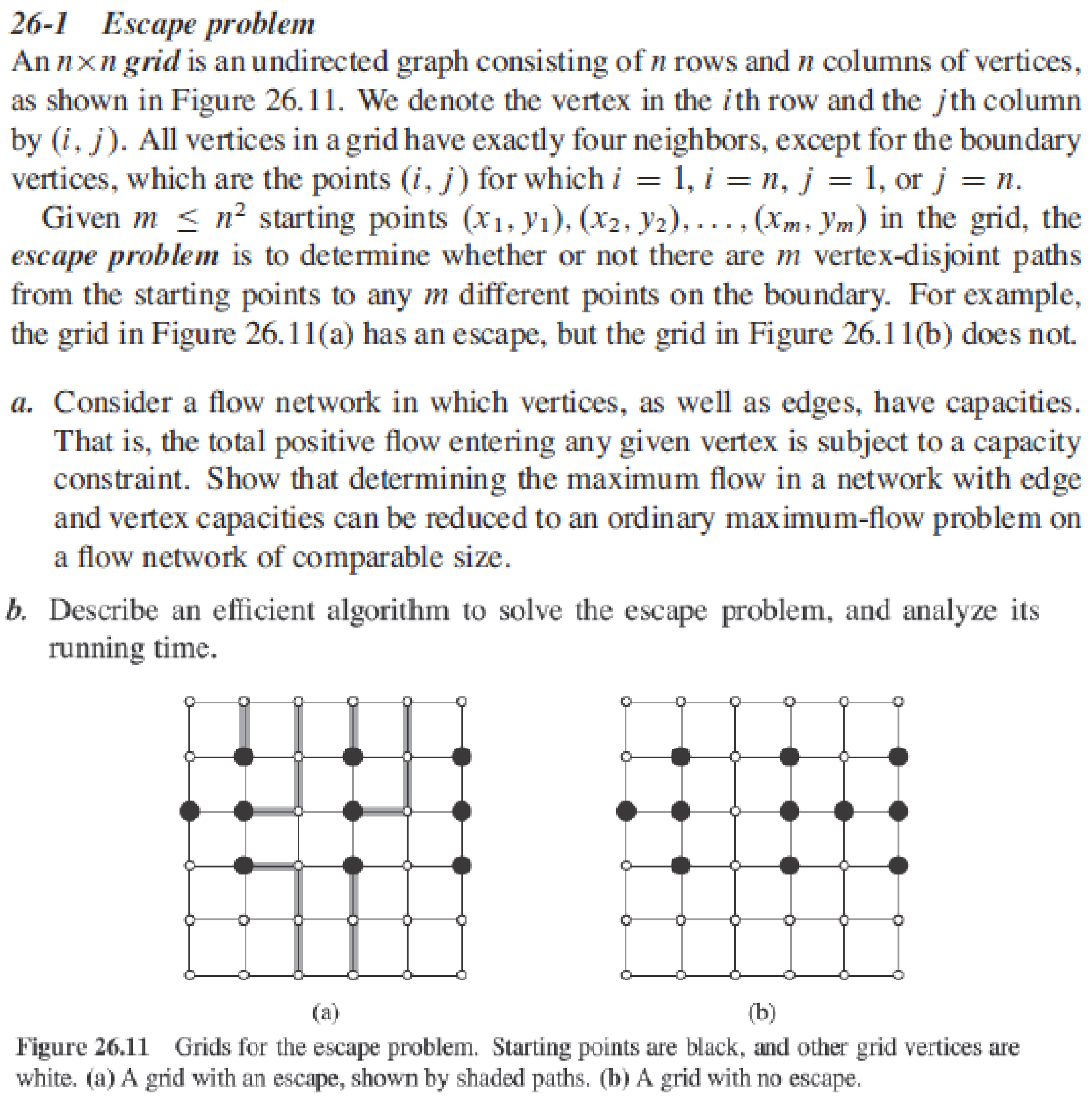
2. Problem 26-1 (pages 760—761)

a.

- How to construct the new flow network?
- What is the **size** of the new flow network?
- Why can the new flow network handle vertex capacities?

b.

- How to construct the flow network?
- How to obtain escape paths?
- Why is your algorithm correct?
- What is the time complexity? (in terms of n)



26-1 Escape problem

An $n \times n$ ***grid*** is an undirected graph consisting of n rows and n columns of vertices, as shown in Figure 26.11. We denote the vertex in the ith row and the jth column by (i, j). All vertices in a grid have exactly four neighbors, except for the boundary vertices, which are the points (i, j) for which $i = 1$, $i = n$, $j = 1$, or $j = n$.

Given $m \le n^2$ starting points $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ in the grid, the ***escape problem*** is to determine whether or not there are m vertex-disjoint paths from the starting points to any m different points on the boundary. For example, the grid in Figure 26.11(a) has an escape, but the grid in Figure 26.11(b) does not.

a. Consider a flow network in which vertices, as well as edges, have capacities. That is, the total positive flow entering any given vertex is subject to a capacity constraint. Show that determining the maximum flow in a network with edge and vertex capacities can be reduced to an ordinary maximum-flow problem on a flow network of comparable size.

b. Describe an efficient algorithm to solve the escape problem, and analyze its running time.

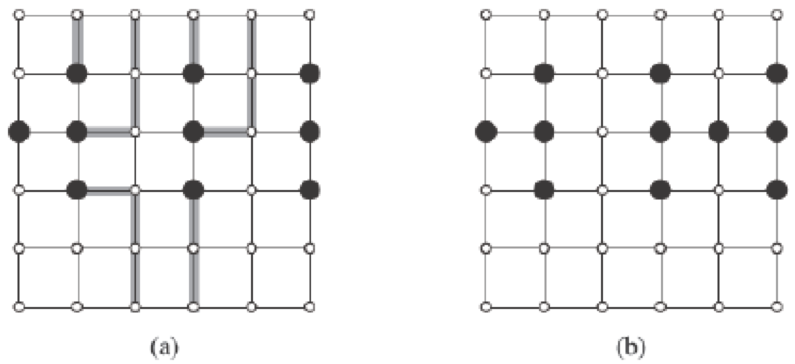


Figure 26.11 Grids for the escape problem. Starting points are black, and other grid vertices are white. **(a)** A grid with an escape, shown by shaded paths. **(b)** A grid with no escape.

3. Problem 26-3 (pages 761—762)

a.

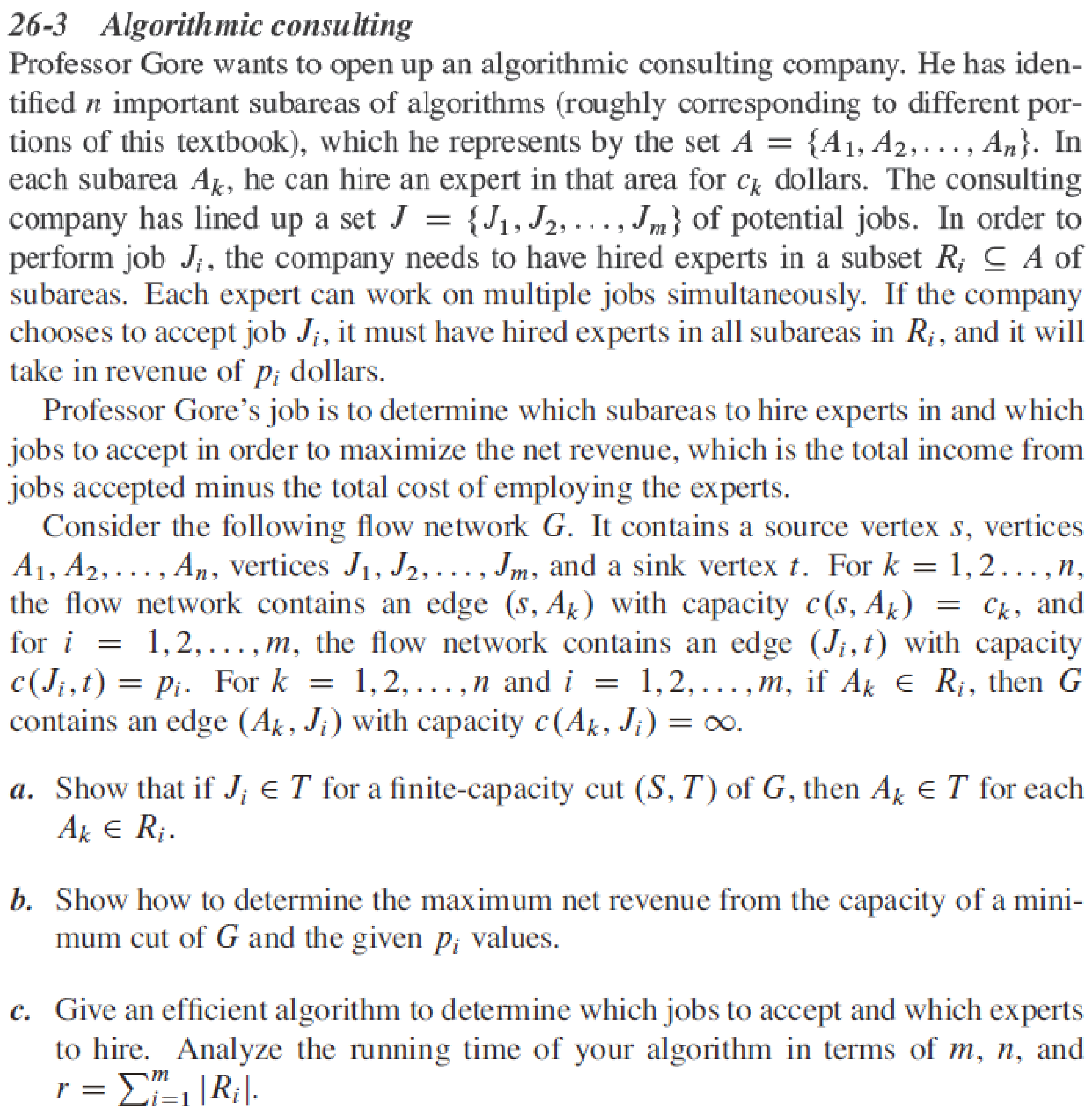
- Prove the statement

b.

- How to determine max net revenue
- Net revenue = ?
  (in terms of $c(S,T)$ and p_i)
- Briefly explain the correctness

c.

- Describe your algorithm
- Running time? (in terms of m, n, r)



26-3 Algorithmic consulting

Professor Gore wants to open up an algorithmic consulting company. He has identified n important subareas of algorithms (roughly corresponding to different portions of this textbook), which he represents by the set $A = \{A_1, A_2, \ldots, A_n\}$. In each subarea A_k, he can hire an expert in that area for c_k dollars. The consulting company has lined up a set $J = \{J_1, J_2, \ldots, J_m\}$ of potential jobs. In order to perform job J_i, the company needs to have hired experts in a subset $R_i \subseteq A$ of subareas. Each expert can work on multiple jobs simultaneously. If the company chooses to accept job J_i, it must have hired experts in all subareas in R_i, and it will take in revenue of p_i dollars.

Professor Gore's job is to determine which subareas to hire experts in and which jobs to accept in order to maximize the net revenue, which is the total income from jobs accepted minus the total cost of employing the experts.

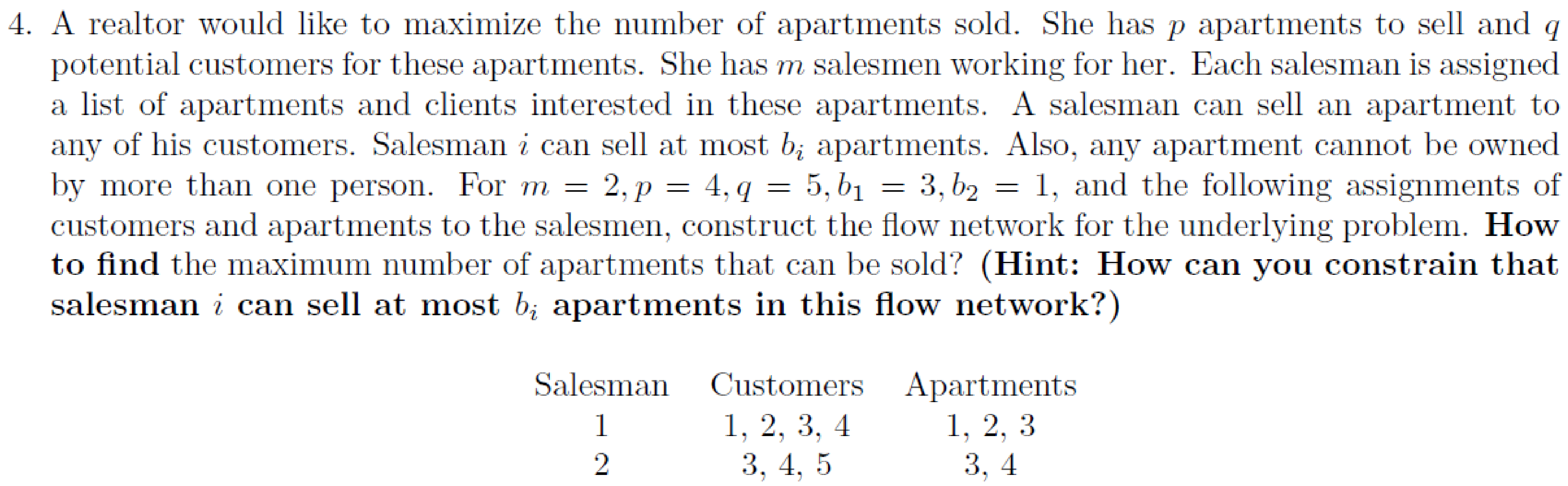
Consider the following flow network G. It contains a source vertex s, vertices $A_1, A_2, \ldots, A_n$, vertices $J_1, J_2, \ldots, J_m$, and a sink vertex t. For $k = 1, 2 \ldots, n$, the flow network contains an edge (s, A_k) with capacity $c(s, A_k) = c_k$, and for $i = 1, 2, \ldots, m$, the flow network contains an edge (J_i, t) with capacity $c(J_i, t) = p_i$. For $k = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, m$, if $A_k \in R_i$, then G contains an edge (A_k, J_i) with capacity $c(A_k, J_i) = \infty$.

a. Show that if $J_i \in T$ for a finite-capacity cut (S, T) of G, then $A_k \in T$ for each $A_k \in R_i$.

b. Show how to determine the maximum net revenue from the capacity of a minimum cut of G and the given p_i values.

c. Give an efficient algorithm to determine which jobs to accept and which experts to hire. Analyze the running time of your algorithm in terms of m, n, and $r = \sum_{i=1}^{m} |R_i|$.

4. Apartment Sale



4. A realtor would like to maximize the number of apartments sold. She has p apartments to sell and q potential customers for these apartments. She has m salesmen working for her. Each salesman is assigned a list of apartments and clients interested in these apartments. A salesman can sell an apartment to any of his customers. Salesman i can sell at most b_i apartments. Also, any apartment cannot be owned by more than one person. For $m = 2, p = 4, q = 5, b_1 = 3, b_2 = 1$, and the following assignments of customers and apartments to the salesmen, construct the flow network for the underlying problem. **How to find** the maximum number of apartments that can be sold? **(Hint: How can you constrain that salesman i can sell at most b_i apartments in this flow network?)**

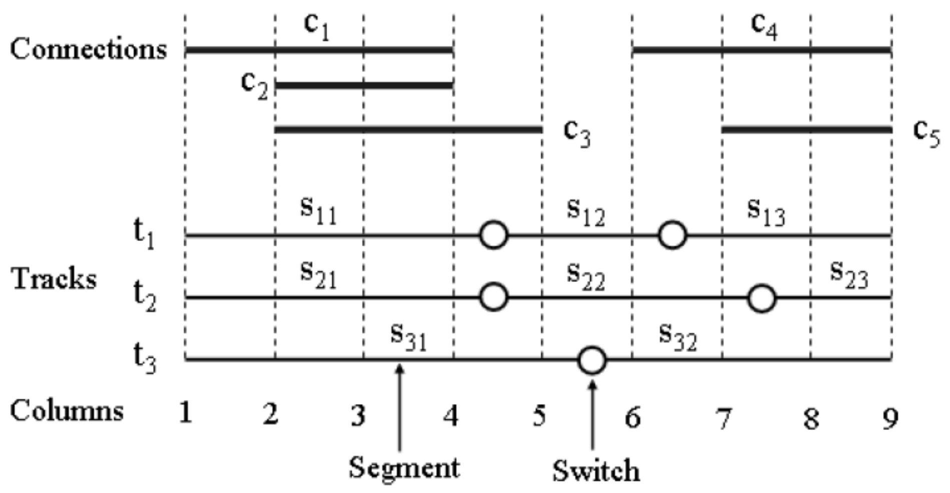
Salesman	Customers	Apartments
1	1, 2, 3, 4	1, 2, 3
2	3, 4, 5	3, 4

- Show the flow network for the given instance (you don't need to find the maximum flow)
  - Only edge capacities are allowed; no vertex capacities!
  - Clearly label each edge with its capacity
- How to obtain the maximum number of apartments that can be sold?

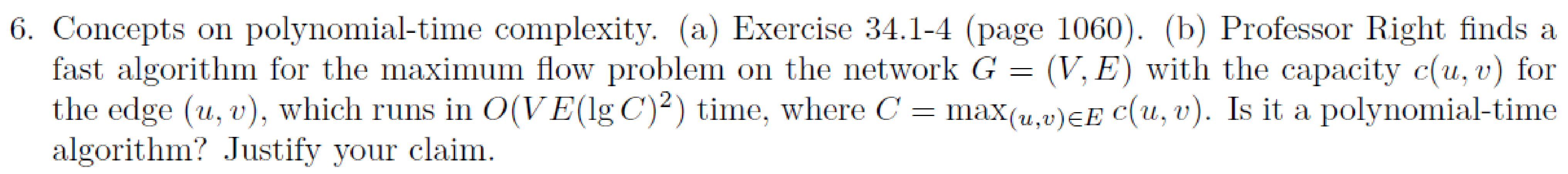
5. FPGA One-segment Routing

5. The figure below shows a segmented routing structure in a row-based field-programmable gate array (FPGA). There are five connections, $c_1, c_2, \ldots, c_5$, to be routed on three segmented tracks, t_1, t_2, and t_3, with eight segments $s_{11}, s_{12}, \ldots, s_{32}$ in the row-based FPGA. A track can be partitioned into a set of segments by using switches. If a switch incident on two adjacent segments is "ON", then the two segments are electrically connected; otherwise, the two segments can be used independently. You are asked to route (place) the five connections on the three segmented tracks. Suppose each connection can use at most one segment for routing, i.e., 1-segment routing. In other words, a connection c_k of the column span $[l_k, r_k]$ is said to be routed on a segment s_{ij} of track t_i if $c_k = [l_k, r_k]$ is placed within the column span of s_{ij}. For example, $c_3 = [2, 5]$ can be routed on segment s_{31} of track t_3 (which consumes only one segment) while it cannot route on track t_1 or t_2 (which would have consumed two segments, thus violating the constraint of 1-segment routing). Give an efficient algorithm to solve the 1-segment routing problem. What is the time complexity of your algorithm?

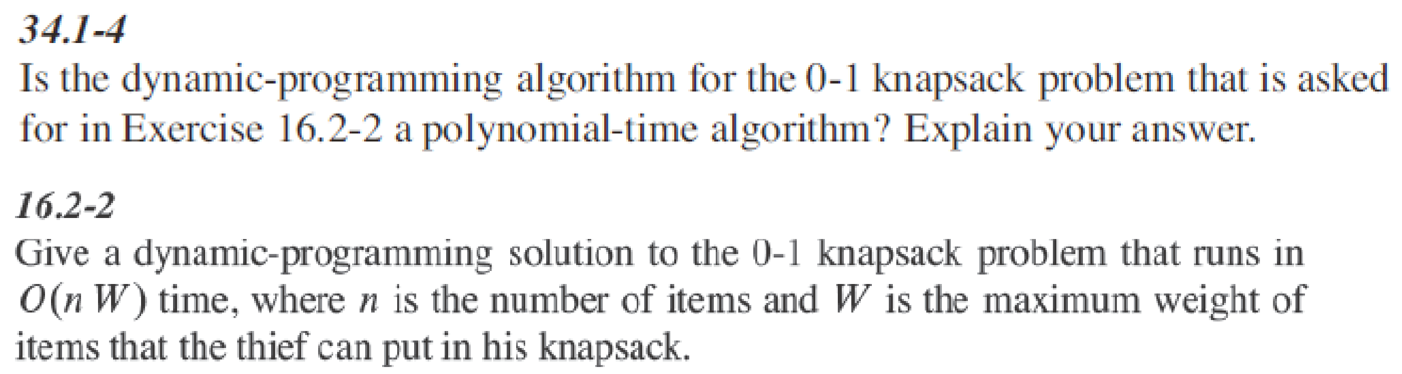
- How to construct the flow network
- Show how you handle the given instance
- Explain how to obtain a routing solution
- Analyze the time complexity



6. Concepts on Polynomial-time Complexity



6. Concepts on polynomial-time complexity. (a) Exercise 34.1-4 (page 1060). (b) Professor Right finds a fast algorithm for the maximum flow problem on the network $G = (V, E)$ with the capacity $c(u, v)$ for the edge (u, v), which runs in $O(VE(\lg C)^2)$ time, where $C = \max_{(u,v)\in E} c(u, v)$. Is it a polynomial-time algorithm? Justify your claim.



34.1-4

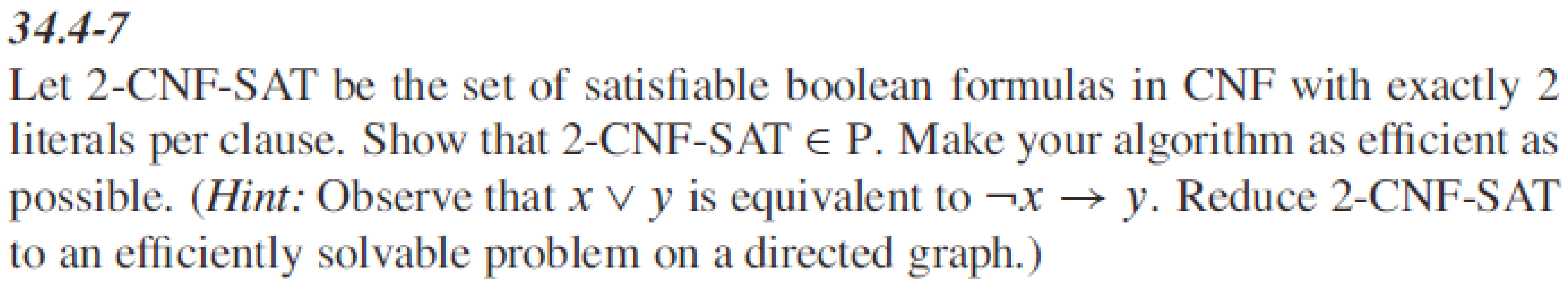
Is the dynamic-programming algorithm for the 0-1 knapsack problem that is asked for in Exercise 16.2-2 a polynomial-time algorithm? Explain your answer.

16.2-2

Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

- Yes or No?
- Explain why (Hint: input size)

7. Exercise 34.4-7 (page 1086)



34.4-7

Let 2-CNF-SAT be the set of satisfiable boolean formulas in CNF with exactly 2 literals per clause. Show that 2-CNF-SAT $\in$ P. Make your algorithm as efficient as possible. (*Hint:* Observe that $x \vee y$ is equivalent to $\neg x \rightarrow y$. Reduce 2-CNF-SAT to an efficiently solvable problem on a directed graph.)

- What is your polynomial-time algorithm to solve the 2-CNF-SAT problem?
  - Graph construction, …
- What is the running time of your algorithm?
  - Must be polynomial-time
  - Make it as efficient as possible
  - Should be in terms of **#variables** n and **#clauses** m
- Prove the correctness of your algorithm

8. Problem 34-1 (pages 1101—1102)

a.

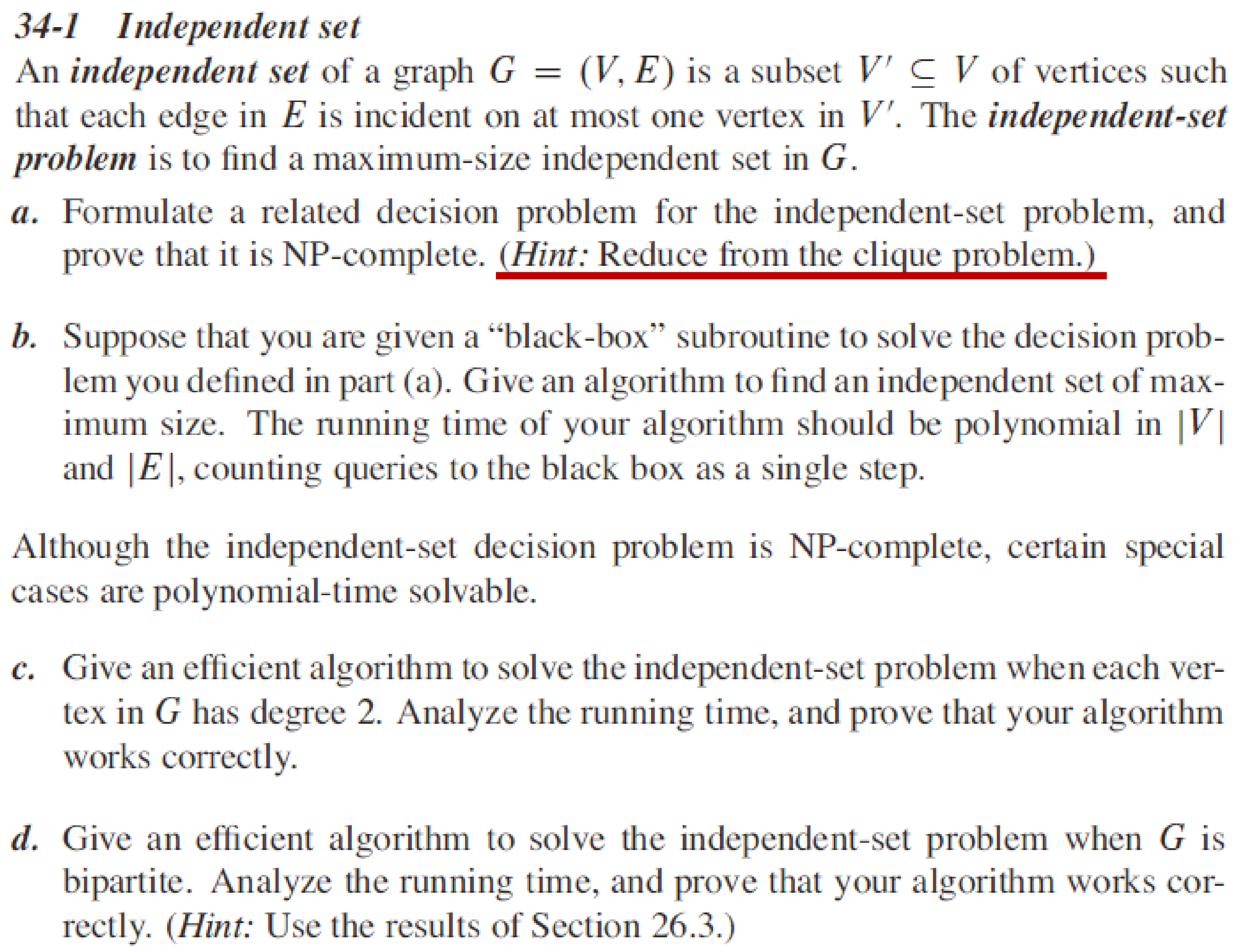
- Decision problem?
- Prove IS is NPC

b.

- Querying “black box” takes $O(1)$ time
- Describe your algorithm
- Analyze the running time

c. & d.

- Describe your algorithm
- Analyze the running time
- Prove the correctness



34-1 Independent set

An ***independent set*** of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in E is incident on at most one vertex in V'. The ***independent-set problem*** is to find a maximum-size independent set in G.

a. Formulate a related decision problem for the independent-set problem, and prove that it is NP-complete. (*Hint:* Reduce from the clique problem.)

b. Suppose that you are given a “black-box” subroutine to solve the decision problem you defined in part (a). Give an algorithm to find an independent set of maximum size. The running time of your algorithm should be polynomial in $|V|$ and $|E|$, counting queries to the black box as a single step.

Although the independent-set decision problem is NP-complete, certain special cases are polynomial-time solvable.

c. Give an efficient algorithm to solve the independent-set problem when each vertex in G has degree 2. Analyze the running time, and prove that your algorithm works correctly.

d. Give an efficient algorithm to solve the independent-set problem when G is bipartite. Analyze the running time, and prove that your algorithm works correctly. (*Hint:* Use the results of Section 26.3.)

9. Problem 34-3 (pages 1103—1104) (1/2)

a.

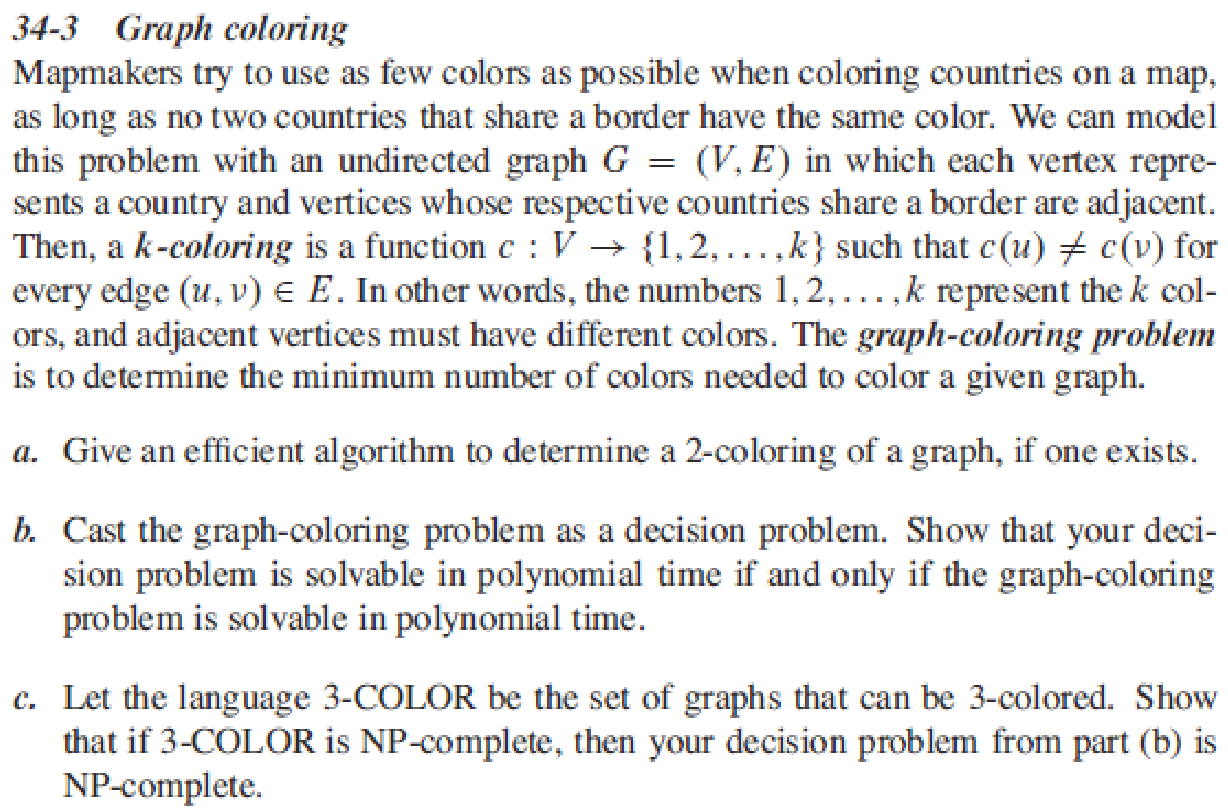
- Determine whether a 2-coloring exists or not?
- What is the time complexity?

b.

- What is the decision problem?
- Prove “if and only if”

c.

- Prove the statement



34-3 Graph coloring

Mapmakers try to use as few colors as possible when coloring countries on a map, as long as no two countries that share a border have the same color. We can model this problem with an undirected graph $G = (V, E)$ in which each vertex represents a country and vertices whose respective countries share a border are adjacent. Then, a ***k-coloring*** is a function $c : V \to \{1, 2, \ldots, k\}$ such that $c(u) \neq c(v)$ for every edge $(u, v) \in E$. In other words, the numbers $1, 2, \ldots, k$ represent the k colors, and adjacent vertices must have different colors. The ***graph-coloring problem*** is to determine the minimum number of colors needed to color a given graph.

a. Give an efficient algorithm to determine a 2-coloring of a graph, if one exists.

b. Cast the graph-coloring problem as a decision problem. Show that your decision problem is solvable in polynomial time if and only if the graph-coloring problem is solvable in polynomial time.

c. Let the language 3-COLOR be the set of graphs that can be 3-colored. Show that if 3-COLOR is NP-complete, then your decision problem from part (b) is NP-complete.

9. Problem 34-3 (pages 1103—1104) (2/2)

d.

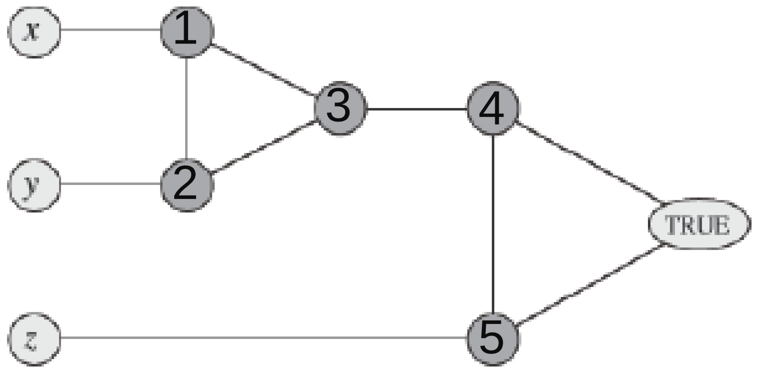
- Prove the two statements

e.

- Prove "if and only if"
- Label the nodes if necessary

f.

- Reduce from 3SAT



To prove that 3-COLOR is NP-complete, we use a reduction from 3-CNF-SAT. Given a formula ϕ of m clauses on n variables $x_1, x_2, \ldots, x_n$, we construct a graph $G = (V, E)$ as follows. The set V consists of a vertex for each variable, a vertex for the negation of each variable, 5 vertices for each clause, and 3 special vertices: TRUE, FALSE, and RED. The edges of the graph are of two types: "literal" edges that are independent of the clauses and "clause" edges that depend on the clauses. The literal edges form a triangle on the special vertices and also form a triangle on x_i, $\neg x_i$, and RED for $i = 1, 2, \ldots, n$.

d. Argue that in any 3-coloring c of a graph containing the literal edges, exactly one of a variable and its negation is colored $c(\text{TRUE})$ and the other is colored $c(\text{FALSE})$. Argue that for any truth assignment for ϕ, there exists a 3-coloring of the graph containing just the literal edges.

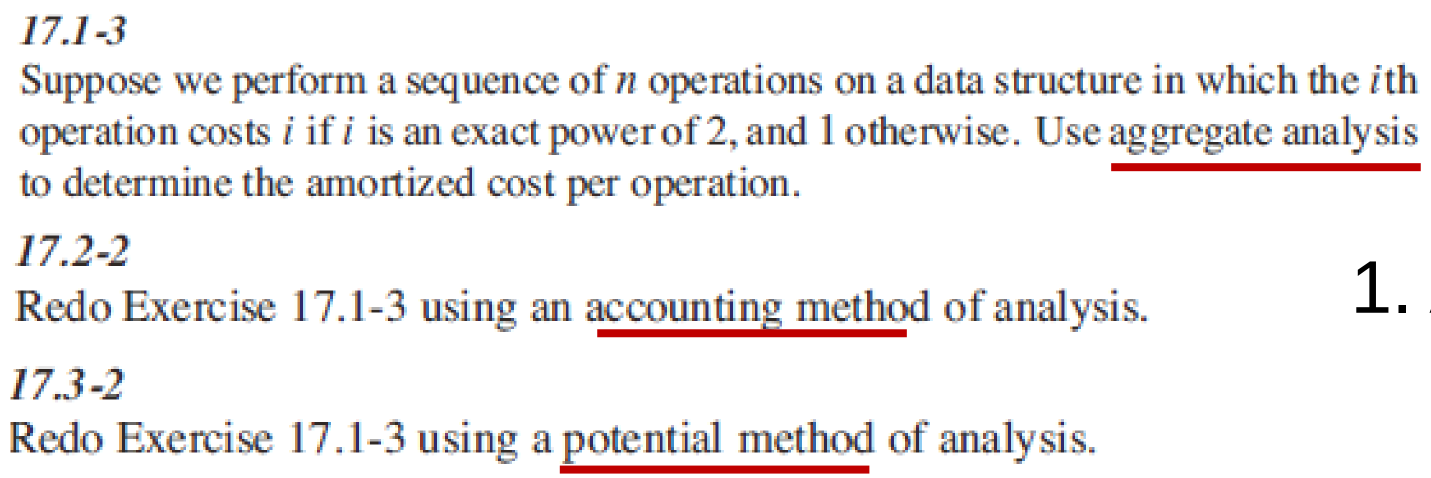
The widget shown in Figure 34.20 helps to enforce the condition corresponding to a clause $(x \vee y \vee z)$. Each clause requires a unique copy of the 5 vertices that are heavily shaded in the figure; they connect as shown to the literals of the clause and the special vertex TRUE.

e. Argue that if each of x, y, and z is colored $c(\text{TRUE})$ or $c(\text{FALSE})$, then the widget is 3-colorable if and only if at least one of x, y, or z is colored $c(\text{TRUE})$.

f. Complete the proof that 3-COLOR is NP-complete.

Figure 34.20 The widget corresponding to a clause $(x \vee y \vee z)$, used in Problem 34 3.

10. Exercise 17.1-3 (page 456), 17.2-2 (page 459), 17.3-2 (page 462)



17.1-3

Suppose we perform a sequence of n operations on a data structure in which the ith operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis to determine the amortized cost per operation.

17.2-2

Redo Exercise 17.1-3 using an accounting method of analysis.

17.3-2

Redo Exercise 17.1-3 using a potential method of analysis.

- Use the specified method
- $i = 1, 2, \ldots, n$
- Exact power of 2: $1, 2, 4, 8, 16, 32, \ldots$

1. Aggregate analysis

$$T(n)/n$$

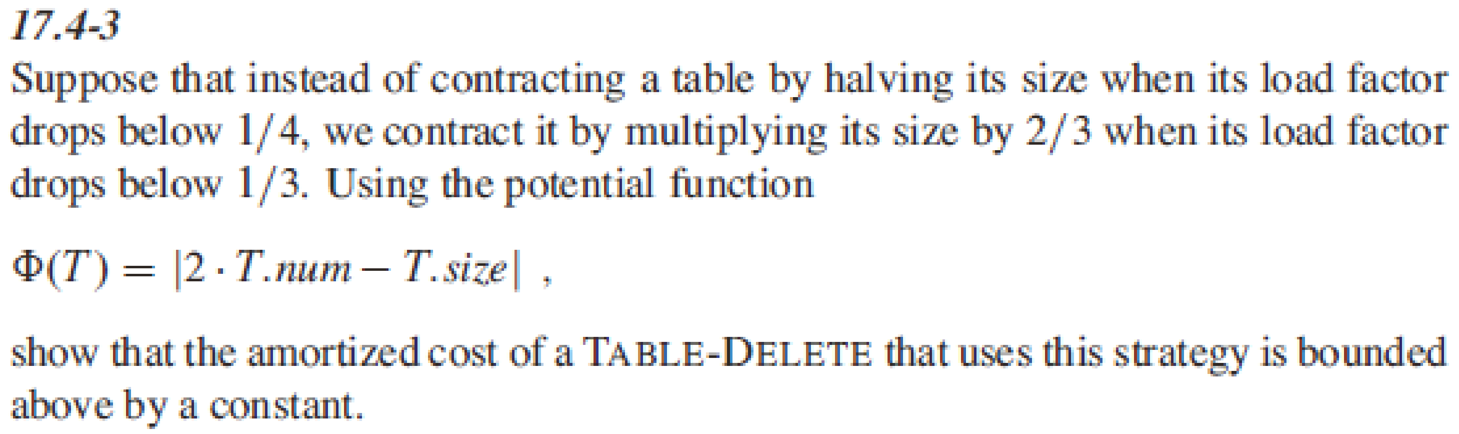
2. Accounting method

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

3. Potential method

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$

11. Exercise 17.4-3 (page 471)



17.4-3

Suppose that instead of contracting a table by halving its size when its load factor drops below $1/4$, we contract it by multiplying its size by $2/3$ when its load factor drops below $1/3$. Using the potential function

$$\Phi(T) = |2 \cdot T.num - T.size| \ ,$$

show that the amortized cost of a TABLE-DELETE that uses this strategy is bounded above by a constant.

- Show all the possible cases
  - Bounded by a constant
- Suppose that table size is always an exact power of 3
- Hint: triangle inequality $|\boldsymbol{a}+\boldsymbol{b}| \leq |\boldsymbol{a}| + |\boldsymbol{b}|$

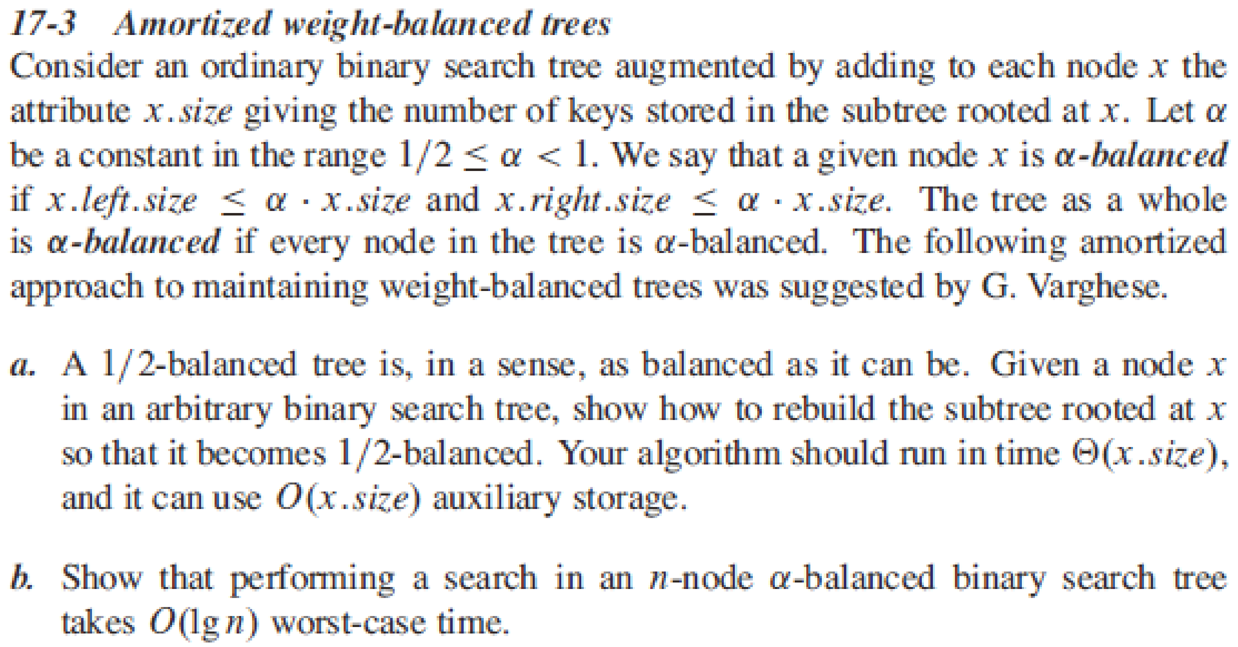
12. Problem 17-3 (pages 473—474) (1/2)

a.

- Describe your algorithm
- Analyze the running time & auxiliary storage
- Explain why the rebuilt tree is 1/2-balanced

b.

- What is the maximum possible tree height?



17-3 Amortized weight-balanced trees

Consider an ordinary binary search tree augmented by adding to each node x the attribute $x.size$ giving the number of keys stored in the subtree rooted at x. Let α be a constant in the range $1/2 \le \alpha < 1$. We say that a given node x is ***α-balanced*** if $x.left.size \le \alpha \cdot x.size$ and $x.right.size \le \alpha \cdot x.size$. The tree as a whole is ***α-balanced*** if every node in the tree is α-balanced. The following amortized approach to maintaining weight-balanced trees was suggested by G. Varghese.

a. A $1/2$-balanced tree is, in a sense, as balanced as it can be. Given a node x in an arbitrary binary search tree, show how to rebuild the subtree rooted at x so that it becomes $1/2$-balanced. Your algorithm should run in time $\Theta(x.size)$, and it can use $O(x.size)$ auxiliary storage.

b. Show that performing a search in an n-node α-balanced binary search tree takes $O(\lg n)$ worst-case time.

12. Problem 17-3 (pages 473—474) (2/2)

c.

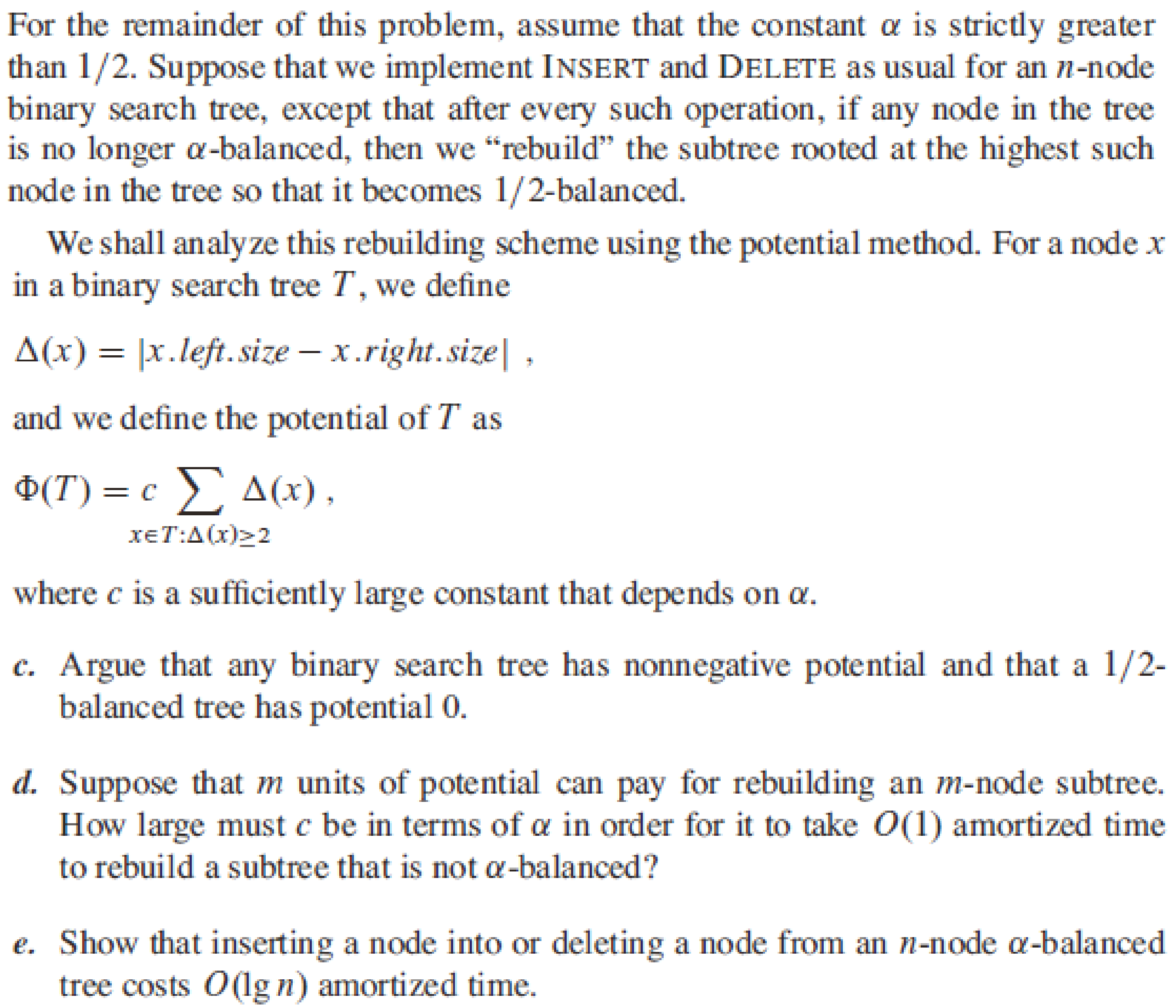
- Argue the **two** statements
- Show that $\Delta(x) \le 1$ for a 1/2-balanced tree

d.

- $c =?$ (in terms of α)
- Show your steps
  - $\hat{c_i} = c_i + \Phi(T_i) - \Phi(T_{i-1})$
    $\le const.$

e.

- You may use the results in previous parts



For the remainder of this problem, assume that the constant α is strictly greater than $1/2$. Suppose that we implement INSERT and DELETE as usual for an n-node binary search tree, except that after every such operation, if any node in the tree is no longer α-balanced, then we "rebuild" the subtree rooted at the highest such node in the tree so that it becomes $1/2$-balanced.

We shall analyze this rebuilding scheme using the potential method. For a node x in a binary search tree T, we define

$$\Delta(x) = |x.left.size - x.right.size| \ ,$$

and we define the potential of T as

$$\Phi(T) = c \sum_{x \in T : \Delta(x) \ge 2} \Delta(x) \ ,$$

where c is a sufficiently large constant that depends on α.

c. Argue that any binary search tree has nonnegative potential and that a $1/2$-balanced tree has potential 0.

d. Suppose that m units of potential can pay for rebuilding an m-node subtree. How large must c be in terms of α in order for it to take $O(1)$ amortized time to rebuild a subtree that is not α-balanced?

e. Show that inserting a node into or deleting a node from an n-node α-balanced tree costs $O(\lg n)$ amortized time.

13. DIY

- For this problem, you are asked to design a problem set related to Chapter(s) 17, 26, and/or 34, and give a sample solution to your problem set. Grading on this problem will be based upon the *quality* of the designed problem as well as the *correctness* of your sample solution.

Notice

- Do not trust the answers on-line
  - Some of them are wrong!!
  - If you submit the on-line wrong answers, ……
- Figure out the answers by yourself
  - Do not copy and paste
  - Write your answers in your words
- If you have any questions, feel free to contact TAs
  - 徐晨皓 *r07943107@ntu.edu.tw*
  - 蔡宇傑 *r07943111@ntu.edu.tw*
  - 鄒建澔 *d08943011@ntu.edu.tw*

Thank You!

National Taiwan University