算法试题

试　　题

A 卷

（考试时间：120 分钟）

班级 _____ 学号_____ 姓名 _____ 任课教师 霍红卫

题号	一	二	三	四	五	六	七	总分
得分								

Problem 1. Bubble-Sort (20 points)

The following is the pseudocode for Bubble-Sort. The length of the input array A is denoted by n. Note that array A is indexed from 1 to n.

```
Bubble-Sort(A)
1  for i ← 1 to n
2     do for j ← 1 to n–1
3           do if A[j] > A[j + 1]
4                 swap elements A[j] and A[j + 1]
```

(a) In terms of n, how many times is line 3 executed in the worst case? What is the worst-case asymptotic running time for Bubble-Sort? (Give the tightest bound possible.)

(b) In terms of n, how many times is line 3 executed in the best case? What is the best-case asymptotic running time for Bubble-Sort? (Give the tightest bound possible.)

(c) We now modify line 2 as follows:

for $j \leftarrow 1$ **to** $n - i$

Brief explain why this modification does not change the behavior of the algorithm.

(d) Now how many times is line 3 executed in the worst case? In the best case? Does this modification change the best-case or the worst-case asymptotic running time of Bubble-Sort?

Problem 2. (8 points, two parts, 4 points each)

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

(a) $T(n) = T(9n/10) + n$.

(b) $T(n) = 4T(n/2) + n^2 \lg n$.

Problem 3. (8 points)

Given a sorted array A of n distinct integers, some of which may be negative, give an algorithm to find an index i such that $1 \leq i \leq n$ and $A[i] = i$ provided such an index exists. If there are many such indices, the algorithm can return any one of them.

Problem 4. Interval-graph coloring problem (10 points)

Suppose that we have a set of activities to schedule among a large number of lecture halls. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall.
(This is also known as the interval-graph coloring problem. We can create an interval graph whose vertices are the given activities and whose edges connect incompatible activities. The smallest number of colors required to color every vertex so that no two adjacent vertices are given the same color corresponds to finding the fewest lecture halls needed to schedule all of the given activities.)

Problem 5. (7 points)

Give a simple example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces incorrect answers. Briefly state why.

Problem 6. (7 points)

Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T(n/2) + n$. Use the substitution method to verify your answer.

试题答案

A 卷

西安电子科技大学计算机学院霍红卫

Problem 1. (20 points)

Solution:

(a) Line 3 is executed $n(n\text{-}1)$ times on any input of length n. Thus, the worst-case running time for BUBBLE-SORT is $\Theta(n^2)$ since there are 6 lines and each line is executed at most $n(n\text{-}1)$ times.

(b) Line 3 is also executed $n(n\text{-}1)$ times in the best case, since it is executed this many times on any input of length n. Thus, the best-case running time of BUBBLE-SORT remains $\Theta(n^2)$. (This is in contrast to INSERTSION-SORT, for which the best-case and the worst-case running time differ by a factor of n.)

(c) Note that after the ith executed of the **for** loop in lines 2-5, the ith largest element in the array A is in the correct place, i.e. the place it should be in the sorted ordering. Thus, on the i+1th execution of this **for** loop, we only need to continue sorting the first $n\text{-}i$ elements.

(d) Line 3 is now executed $\sum_{i=1..n}(n-i) = n(n\text{-}1)/2$ times for any input for length n. This does not change the best-case or worst-case asymptotic running times.

Problem 2. (8 points, two parts, 4 points each)

Solution:

(a) $T(n)= \Theta(n)$. We have $a = 1$, $b = 10/9$, $f(n) = n$, so $n^{\log_b a} = n^0 = 1$. Since $f(n) = n = \Omega(n^{0+\varepsilon})$, case 3 of the Master Theorem applies if we can show the regularity condition holds. For all n, $af(n/b) = (9n)/10 \le (9/10)n = c\,f(n)$ for $c = 9/10$. Therefore, case 3 tells us that $T(n) = \Theta(n)$.

(b) $T(n)= \Theta(n^2\lg^2 n)$. We have $a = 4$, $b = 2$, $f(n) = n^2\lg n$, so $n^{\log_b a} = n^2$. $f(n)$ is asymptotically larger than $n^{\log_b a}$, but not polynomial larger. The ratio $\lg n$ is asymptotically less than n^{ε} for any positive constant ε. Thus, the Master Theorem doesn't apply here.

Problem 3. (8 points)

Solution: The key observation is that if $A[j] > j$ and $A[i] = i$, then $i < j$. Similarly if $A[j] < j$ and $A[i] = i$, then $i > j$. So if we look at the middle element of the array, then half of the array can be eliminated. The algorithm below (INDEX-SEARCH) is similar to binary search and runs in $\Theta(\log n)$ time. It returns -1 if there is no answer.

```
INDEX-SEARCH(A, b,e )
1  if (e > b)
2     return -1
3     m ←⌈(e+b)/2⌉
4     if A[m] = m
5        then return m
6     if A[m] > m
7        then return INDEX-SEARCH(A, b, m)
8        else return INDEX-SEARCH(A,m , e)
```

Problem 4. (10 points)

Solution: Assume that the activities are in the form $A[i] = [s_i, f_i]$ for $1 \le i \le n$ and that the lecture halls are 1, 2, ..., r. As in the simpler (one hall) activities-scheduling problem, assume that $f_i \le f_j$ for all $1 \le i < j \le n$. The idea is to assign to $A[i]$ hall $H[i]$ ($1 \le i \le n$), which is taken to be the available hall with the least label. To keep track, we let $E[i]$ ($1 \le i \le r$) be the ending time of the last activity schedule into hall i.

```
LECTURE-HALL-SCHEDULER(s, f)
1  for i ← 1 to r
2     do E[i] ← 0
3  for i ← 1 to n
4     do j ← min{k | E[k] ≤ s_i}
5        H[i] ← j
6        E[j] ← f_i
7  return H
```

A straightforward implementation of this algorithm has time complexity $O(nr)$. Since there is certainly no reason to have more than n halls available, $r = O(n)$, so the algorithm is at worst quadratic in the size of the problem. There are enhancements possible that drive the cost down to $O(n)$, but require a certain amount of additional theory.

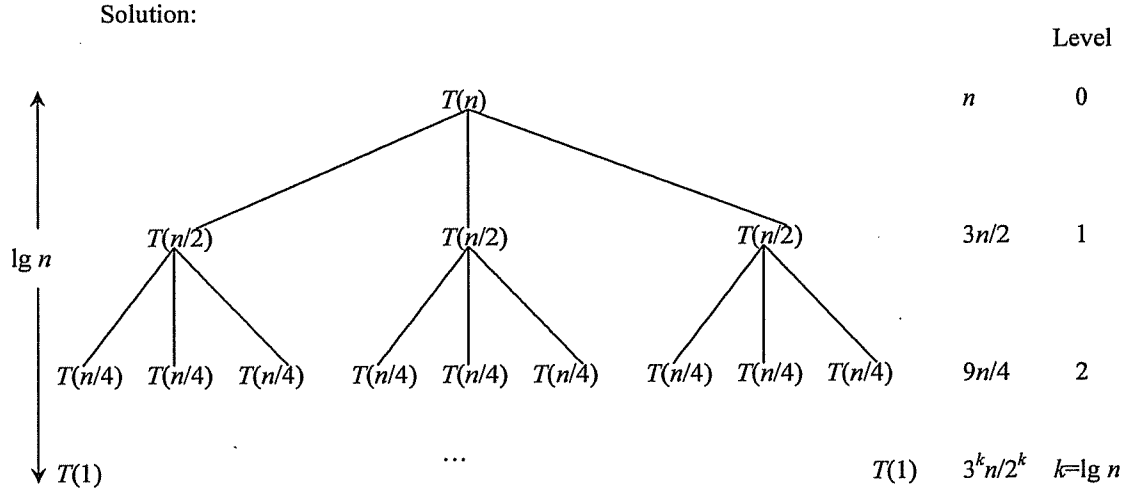
Problem 5. (7 points)

Solution: The graph is on four vertices s, t, u,v , where $w(s, u) = 4$, $w(s, t) = 3$, $w(u, t) = -2$, and $w(t, v) = 1$. Starting from s, we set $d[t] = 3$ and $d[u] = 4$. Therefore t is extracted, so we set $d[v] = d[t] + 1 = 4$. Next v is extracted, and no changes are made to d. Finally u is extracted and we set $d[t] = d[u] + (-2) = 2$, then the algorithm terminates. Note that the shortest path to v is s, u,t , v, and has length 3. However, at the end of the algorithm, $d[v] = 4$ (corresponding to the path s, t, v).

The proof of Theorem 24.6 fails where it claims that $\delta(s, y) \leq \delta(s, u)$ "because y occurs before u on a shortest path from s to u all edge weights are nonnegative." In fact, we see in the above example that this is not the case: the shortest path from s to t is s, u, t and has length 2, but the shortest path from s to u has length 4. Therefore the proof of correctness is no longer sound.

Problem 6. (7 points)

Solution:



Determine an upper bound on $T(n) = 3T(n/2) + n$ using a recursion tree. We have that each node of depth k is bounded by $n/2^k$ and therefore the contribution of each level is at most $(3/2)^k n$. The last level of depth lg n contributes $\Theta(3^{\lg n}) = \Theta(n^{\lg 3})$. Summing up we obtain:

$$
\begin{aligned}
T(n) &= 3T(n/2) + n \\
&\leq n + (3/2)n + (3/2)^2 n + \ldots + (3/2)^{\lg n - 1} n + \Theta(n^{\lg 3}) \\
&= n \sum_{k=0..\lg n-1} (3/2)^k + \Theta(n^{\lg 3}) \\
&= n\,[(3/2)^{\lg n} - 1]/[(3/2) - 1] + \Theta(n^{\lg 3}) \\
&= 2[n\,(3/2)^{\lg n} - n]/[(3/2) - 1] + \Theta(n^{\lg 3}) \\
&= 2n\,(3^{\lg n}/2^{\lg n}) - 2n + \Theta(n^{\lg 3})
\end{aligned}
$$

$= 2\,3^{\lg n} - 2n + \Theta(n^{\lg 3})$
$= 2\,n^{\lg 3} - 2n + \Theta(n^{\lg 3})$
$= \Theta(n^{\lg 3})$

We can prove this by substitution by assuming that $T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor^{\lg 3} - c\lfloor n/2 \rfloor$. We obtain

$T(n) = 3T(\lfloor n/2 \rfloor) + n$
$\le 3c\lfloor n/2 \rfloor^{\lg 3} - c\lfloor n/2 \rfloor + n$
$\le 3c\,n^{\lg 3}/2^{\lg 3} - cn/2 + n$
$\le c\,n^{\lg 3}$,

where the last inequality holds for $c \ge 2$.

2005年

一、Judge

a) If a DP problem satisfied the optimal-substructure property ,then a local optimal solution is globally optimal.

Answer:False.The property which implies that locally optimal solutions are globally optimal is the greedy-choice property.Consider as a counterexample the edit distance problem.This problem has optimal substructure,as was shown on the problem set,however a locally optimal solution .Making the best edit next does not result in a globally optimal solution.

b) Let T be complete binary tree with n nodes. Finding a path from root of T to a given vertex $v \in T$ using breadth-first search takes $O(\log n)$ time.

Answer:False .Breath-first search requires $\Omega(\lg n)$ time.Breadth-first search examines each node in the tree in breadth-first order.The vertex v could well be the last vertex explored (Also ,notice that T is not neccesarily sorted).

c) Radix-SORT works correctly if we sort individual digit with HEAP-SORT instead of COUNT-SORT.

Answer:False. Heapsort is not stable .If the first i-1 digits are sorted in RADIX SORT, sorting the ith digit using stable won't change the order of two items with the same ith digit This cannot be guaranteed with HEAPSORT.

d) There exists a comparison-based sorting algorithm that can sort any 6-element array using at most 9 comparisons.

Answer:A decision tree for soting a 6-element array has to have at least $6!=720$ leaves ,but if we perform at most 9 comparions we can get at most $2^6=512$ leaves which is less than 720.

e) Consider an implementation of Quicksort which always chooses the partition element x to be equal to A[3], where A[1...n] is the array to partition. This implementation of Quicksort runs in O(nlg n) time in the worst case.

Answer:false.If we apply it to the array n,n-1,1,2,3,4...,n-2,we woulc get a running time of $T(n)=T(n-1)+\Theta(n)$,which is $T(n)=\Theta(n^2)$.

f) The following array A is a min-heap.

i. 2 5 9 8 10 13 12 22 50

g) Computing the most frequently occurring element in an array A[1...n] can be done in $O(n \lg n)$ time.

h) The decision-tree lower bound on comparison. Sorting can be used to prove that the number of comparison needed to build a heap of n elements is $\Omega(n \lg n)$ in the worst case.

二、 The following are a few of design strategies we followed in class to solve the problems (18points, 9parts, 2points each)

a) D_P
b) Greedy strategy
c) Divide and conquer
d) Increment method 增量方法.
e) None of the above

For each of the following ,mention which of the above design strategies was used in the following algorithm

a i. Matrix-chain multiplication
d ii. Insertion sort
e iii. Counting sort
c iv. Quicksort
v. Select in the worst-case time
b vi. Huffman codes
b vii. Activity-selection problem
c viii. Merge-sort
a ix. Assembly-line scheduling 装配线调度.

三、 (10points) Using master theorem to solve the following recurrences $T(n)=aT(n/2)+cn^3$. for the three cases a=7,8,9 and a constant c>0.

$a=7,\ b=2,\ n^{\log_b a}$, $f(n)=cn^3$

$n^{\log_2 8}=n^3$

四、 (10points) Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n)=3T(n/2)+n$. Use the substitution method to verify your answer.

五、 Short answers(16 points)

a) Describe the difference between average-case and worst case analysis of algorithms ,and give an example of an algorithm whose average-case running time different from its worst-case running time. quick sort 相邻元素.

b) Suppose that an array A has the property that only adjacent elements might be out of order -ie,if i<j and A[i]>A[j],then j=i+1. Which of insert-order or Mergesort is a better is a better algorithm to sort the elements of A? Justify your choice.

六、 (10 points) find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is <5,10,3,12,5,50,6>. Using the p_{201}.

equation $m[i,j]=\begin{cases}0, if & i=j\\ \min_{i\le k<j}\{\min[i,k]+m[k+1,j]+p_{i-1}p_k p_j\}, if & i<j\end{cases}$

七、 (12 points)

What is an optimal huffman code for the following set frequencies , based on the first 8 Fibonacci numbers?

A:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21

2007年

1.(12points)Using Master' Theorem to solve the following

a.T(n)=T(2n/3)+n

b.T(n)=2T(n/2)+n

c.T(n)=T(n/3)+n^2

2.(12points)For each of the following questions the answers are chosen from

A.Insertion sort

B.Merge sort

C.Heap sort

D.Quicksort

a.Which of the above algorithms are "in place"?

b.Which of the above algorithms are "divde-and-conquer"algorithms?

c.Which of the above algorithms have their worst-case running time?

d.Which of the above algorithms have their best-case running time?

e.Which of the above algorithms have O(nlgn) average-case of worst -case time?

4.(12points)Consider the knapsack problem with the following items with weights and values as follows:

A.50lbs,$200 B.30lbs,$180 C.45lbs,$225 D.25lbs,$200 E.5lbs,$50

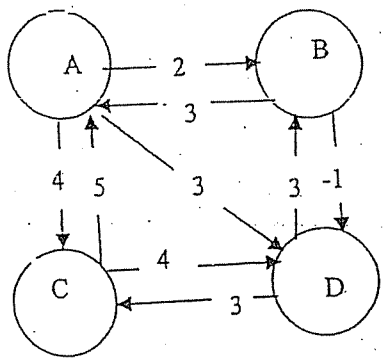
a.Consider the fractional knapsacd problem where the weight constraint is 100lbs. What is the greatest value that can be achieved given the constraint?

b.Now consider the 0-1 knapsack problemwhere the weight constraint is 100lbs. What is the greatest value that can be achieved given the constraint and using the greedy approach?

5.(10points)Find a longest common subsequence of <A,C,G,A,T,C,T> and < G,C,T,A,C,T> please write out both the c[...] and b[...] table values for your solution in the usual format for LCS calculations.

6.(12points)Consider the following input to Counting Sort as shown in the notes
A=<2,7,4,3,5,2,1,3,2,4,5,7>
a.Show the value of the C array after the first step (not including initial step) and the second step of Counting Sort.
b.Show the contents of the B and C arrays after having placed the first three values of A into B.

7.using A as the source ,and each pass relas the edges in the order AB ,AC ,AD,BA,BD,DB,CA,CD .show the d and values after two passes.



8.(10points)What characteristics of a problem do you look for when considering whether or not to use dynamic programming?

9.(8points)Consider the number triangle shown below.Design an algorithm that calculates the highest sum of numbers that can be passed on a route that starts at the top and ends somewhere on the base.Each step can go either diagonally down to the left or diagonally down to the right .For example

```
        7
      3   8
    8   1   0
  2   7   4   4
4   5   2   6   5
```

In the sample above,the route from 7 to 3 to 8 to 7 to 5 produces the highest sum:30.

2007 年

Problem 1. Algorithm Design Techniques (20 points)

The following are a few of the design strategies we followed in class to solve several problems.

(1) Dynamic programming
(2) Greedy strategy
(3) Divide-and-conquer
(4) None of the above

For each of the following problems, mention which of the above design strategies was used in the following algorithm

1. Longest common subsequence algorithm

2. Minimum spanning tree algorithm (Prim's algorithm)
3. Single-source shortest paths (Dijkstra's algorithm)
4. Quicksort
5. Select in the worst-case time
6. Fast Fourier Transform
7. Activity-Selection problem
8. Mergesort
9. Bellman-Ford algorithm
10. Floyd-Warshall algorithm

Problem 2. Using Master' Theorem to solve the following recurrences. You can assume T(n) = 1 for n smaller than some constant in all cases. (18 points, 3 points each)

(a) $T(n) = T(n/7) + 1$
(b) $T(n) = 10\,T(n/3) + n^{1.1}$
(c) $T(n) = 3\,T(n/2) + n \lg n$.

(d) $T(n) = T(n/2) + 2^n$

(e) $T(n) = 3T(n/3) + n$

(f) $T(n) = 3T(n/4) + n \log n$.

Problem 3. (9 points) We will to define a universal hash function family $\mathcal{H} = \{h_1, h_2, h_3\}$ where $U = \{a, b, c\}$ and $h_i : U \rightarrow \{0, 1\}$. Complete the following table to make $\mathcal{H}$ universal and briefly state why.

The function h_1 has already been defined for you:

	a	b	c
h_1	0	0	1
h_2			
h_3			

Problem 4. True or False, and Justify (18 points, 3 points each)
Circle **T** or **F** for each of the following statements, and briefly state why. The better your argument, the higher your grade, but be brief. Your justification is worth more points than your true-or-false designation.

T F The following array A is a Min Heap:
2 5 9 8 10 13 12 22 50

T F An inorder traversal of a Min Heap will output the values in sorted order.

T F Consider a directed graph G in which every edge has a positive edge weight. Suppose you create a new graph G' by replacing the weight of each edge by the negation of its weight in G. For a given source vertex s, you compute all shortest path from s in G' using Dijkstra's algorithm.
True or false: the resulting paths in G' are the longest (i.e., highest cost) simple paths from s in G.

T F Consider the following algorithm for computing the square root of a number x:

SQUARE-ROOT(x)
1 **for** $i = 1,\ldots,x/2$:
2 if $i^2 = x$ then output i.

True or False: This algorithm runs in polynomial time.

T F Dijkstra's algorithm works correctly on graphs with negative-weight edges, as long as there are no negative-weight cycles.

T F RADIX SORT works correctly if we sort each individual digit with HEAPSORT instead of COUNTING SORT.

Problem 5. Short Answer (10 points, 5 points each)

Give brief, but complete, answer to the following questions.

(a) Describe the difference between average-case and worst-case analysis of algorithms, and give an example of an algorithm whose average-case running time is different from its worst-case running time.

(b) Suppose that an array A has the property that only adjacent elements might be out of order -- i.e., if i < j and A[i] > A[j], then j = i+1. Which of INSERTION-SORT or MERGESORT is a better algorithm to sort the elements of A? Justify your choice.

Problem 6. Mode finding (10 points)

Assume that you are given an array A[1...n] of distinct numbers. You are told that the sequence of numbers in the array is unimodal, i.e., there is an index i such that the sequence A[1...i] is increasing (i.e., A[j] < A[j+1] for $1 \leq j < i$-1) and the sequence A[i...n] is decreasing. The index i is called the mode of A.

Show that the mode of A can be found in O(log n) time.

Problem 7. (15 points)

Santa Claus is packing his sleigh with gifts. His sleigh can hold no more than c pounds. He has n different gifts, and he wants to choose a subset of them to pack in his sleigh. Gift i has utility u_i (the amount of happiness gift i induces in some child) and weight w_i. We define the weight and utility of a set of *gifts* as follows:

- The weight of a set of gifts is the *sum* of their weights.
- The utility of a set of gifts is the *product* of their utilities.

For example, if Santa chooses two gifts such that $w_1 = 3$, $u_1 = 4$ and $w_2 = 2$, $u_2 = 2$, then the total weight of this set of gifts is 5 pounds and the total utility of this set of gift is 8. All numbers mentioned are positive integers and for each gift i, $w_i \leq c$. Your job is to devise an algorithm that lets Santa maximize the utility of the set of gifts he packs in his sleigh without exceeding its capacity c.

(a) A greedy algorithm for this problem takes the gifts in order of increasing weight until the sleight can hold no more gifts. Give a small example to demonstrate that the greedy algorithm does not generate an optimal choice of gifts.

(b) Give a recurrence that can be used in a dynamic programming to compute the maximum utility of a set of gifts that Santa can pack in his sleigh. Remember to evaluate the base cases for your recurrence.

2008 年

Problem 1.(11 points)

(a) Illustrate the operation of MAX-HEAPIFY(A, 3) on the array A = <27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0>.

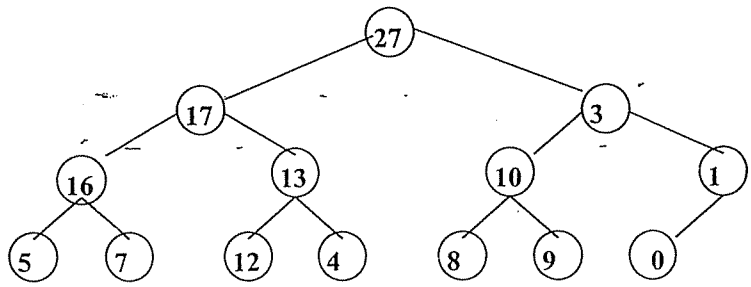


Figure 1

(b) What is the effect of calling MAX-HEAPIFY(A, i) when the element A[i] is larger than its children?

Problem 2. (9 points). Using Master' Theorem to solve the following recurrences:

(a) $T(n) = 4T(n/2) + n$

(b) $T(n) = 4T(n/2) + n^2$

(c) $T(n) = 4T(n/2) + n^3$

Problem 3. (8 points)

Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

Problem 4 (10 points) Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is <5, 10, 3, 12, 5, 50, 6>. Using the equation:

$$m[i,j] = \begin{cases} 0 & \text{if } i = j, \\ \min_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1} p_k p_j\} & \text{if } i < j. \end{cases}$$

Problem 5 (12 points) (a)Give a greedy algorithm for the activity-selection problem; (b)Compute a maximum-size subset of mutually compatible activities for the following activity-selection problem using your algorithm.

i	1	2	3	4	5	6	7	8	9
s_i	1	2	4	1	5	8	9	11	13
f_i	3	5	7	8	9	10	11	14	16

where each activity a_i has a start time s_i and a finish time f_i.

Problem 6 (10 points) Run Dijkstra's algorithm on the directed graph of Figure 2, (a) first using vertex s as the source; (b) and then using vertex z as the source. Show the d and π values and the vertices in set S after each iteration of the while loop.

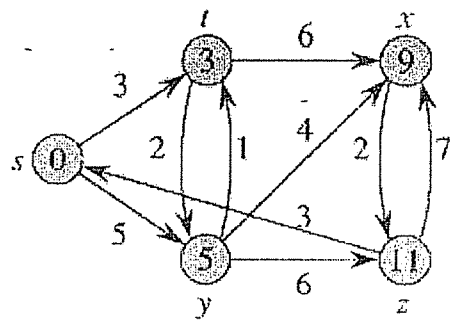


Figure 2