CSCE 221 Cover Page
Homework Assignment #3
Due November 30 to CSNet

First Name: Mitesh Last Name: Patel UIN: 124002210

User Name: patel221881 E-mail address: patel221881@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources		
People	HRBB Peer teacher	TA
Web pages (provide URL)	Listed below	
Printed material	Data Structures and Algorithms Text Book	Discrete Mathematics and its Applications
Other Sources	Lecture Slides	

Resources: http://www.geeksforgeeks.org/find-longest-path-directed-acyclic-graph/
http://stackoverflow.com/questions/1413859/optimizing-dijkstra-for-dense-graph
http://www.informit.com/articles/article.aspx?p=169575&seqNum=3
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
https://en.wikipedia.org/wiki/Heap_(data_structure)

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Your Name Mitesh Patel Date 11-25-15

Homework 3

due November 30

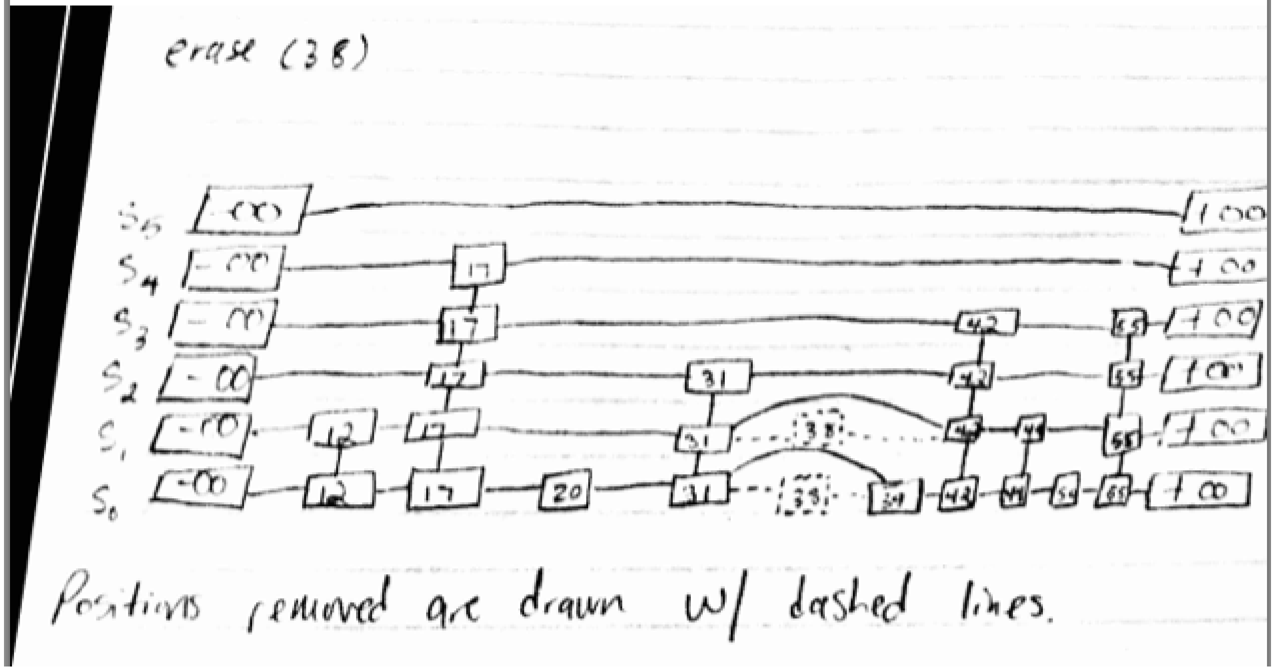
1. (10 points) R-10.17 p. 493
   For the following statements about red-black trees, provide a justification for each true statement and a counterexample for each false one.
   (a) A subtree of a red-black tree is itself a red-black tree
      i. False, since as a counter example consider a tree with a red root, then it is not a red black tree by definition.
   (b) The sibling of an external node is either external or it is red.
      i. True, lets consider a external node that has black siblings then the black depth would be atleast h+1, which violates depth property of black tree that states that all external nodes must have same black depth.
   (c) There is a unique (2,4) tree associated with a given red-black tree.
      i. False, since a 3-node can have different representations in a red black tree.
   (d) There is a unique red-black tree associated with a given (2,4) tree.
      i. True, because we can merge each black node with its red children (0,1,2 red children) to form either a 2-node, 3-node, or 4-node.

2. (10 points) R-10.19 p. 493
   Consider a tree T storing 100,000 entries. What is the worst-case height of T in the following cases?
   n = 100,000
   (a) T is an AVL tree.
      i. $h_{avl} \leq 1.44\log_2(n+1)$
      ii. $h_{avl} \leq 1.44\log_2(100000+1)$
      iii. $h_{avl} \leq 23.9179$
      iv. $= 24$
   (b) T is a (2,4) tree.
      i. $h_{(2,4)} \leq \log_2(n+1)$
      ii. $h_{(2,4)} \leq \log_2(100000+1)$
      iii. $h_{(2,4)} \leq 16.6097$
      iv. $= 17$
   (c) T is a red-black tree.
      i. $h_{RB} \leq 2\log_2(n+1)$
      ii. $h_{RB} \leq 2\log_2(100000+1)$
      iii. $h_{RB} \leq 2(16.6097)$
      iv. $= 33$
   (d) T is a binary search tree.
      i. Worse case is linear so n
      ii. = 100,000

3. (10 points) R-9.16 p. 418

   Draw an example skip list that results from performing the following series of operations on the skip list shown in Figure 9.12: `erase(38)`, `insert(48,x)`, `insert(24,y)`, `erase(55)`. Record your coin flips, as well.

(a)

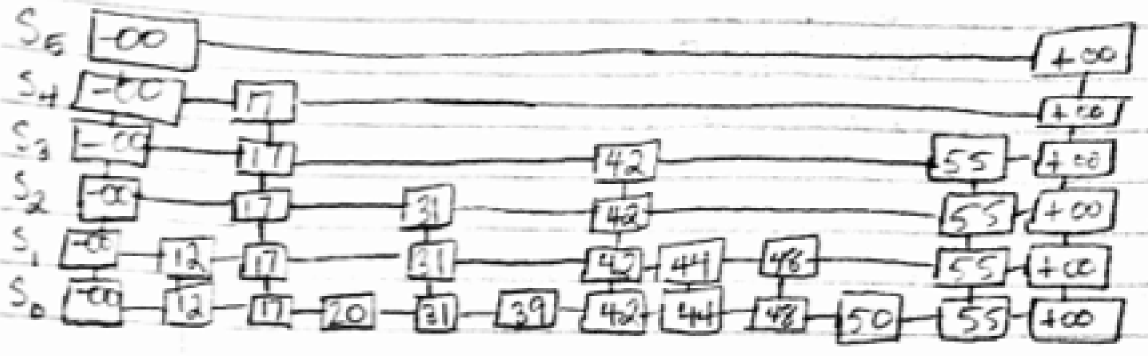


erase (38)

Positions removed are drawn w/ dashed lines.

(b)

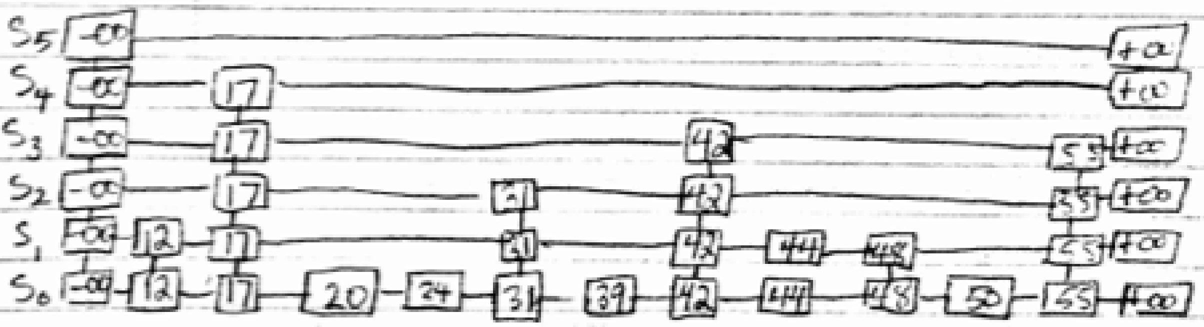
insert (48, x)



S_5 -∞ +∞
S_4 -∞ 17 +∞
S_3 -∞ 17 42 55 +∞
S_2 -∞ 17 31 42 55 +∞
S_1 -∞ 12 17 31 42 44 48 55 +∞
S_0 -∞ 12 17 20 31 39 42 44 48 50 55 +∞

i=1 got heads then tails so insert 48 into S_0, S_1

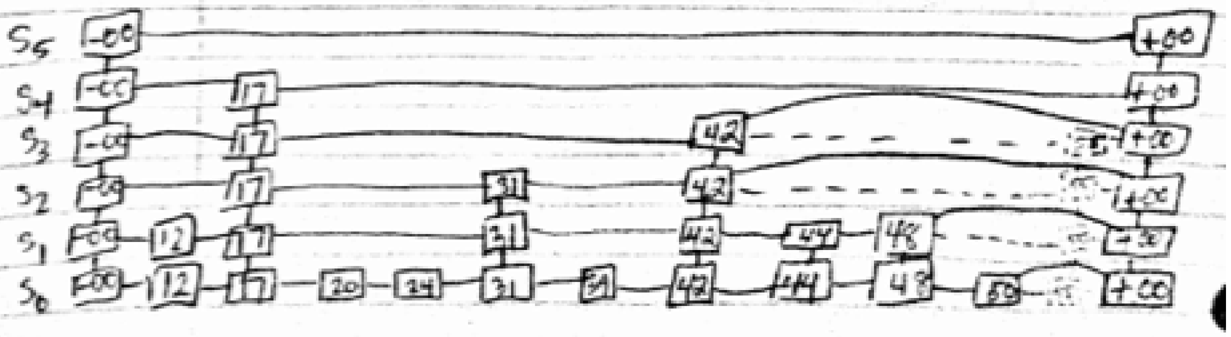
insert (24, y)



S_5 -∞ +∞
S_4 -∞ 17 +∞
S_3 -∞ 17 42 55 +∞
S_2 -∞ 17 31 42 55 +∞
S_1 -∞ 12 17 31 42 44 48 55 +∞
S_0 -∞ 12 17 20 24 31 39 42 44 48 50 55 +∞

i=0 got tails first flip so insert in S_0

Erase(55)



S_5 -∞ +∞
S_4 -∞ 17 +∞
S_3 -∞ 17 42 55 +∞
S_2 -∞ 17 31 42 55 +∞
S_1 -∞ 12 17 31 42 44 48 55 +∞
S_0 -∞ 12 17 20 24 31 39 42 44 48 50 55 +∞

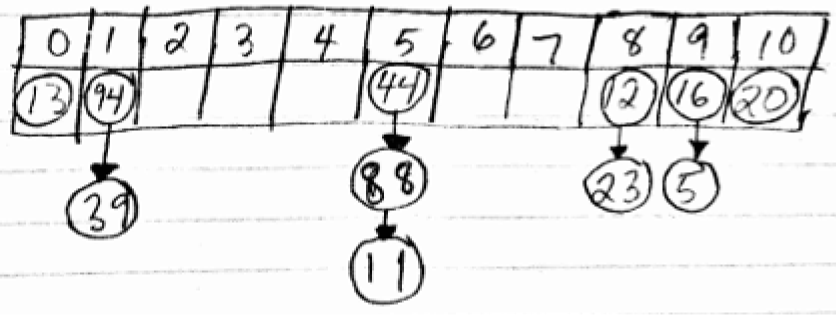
Positions removed are drawn w/ dashed lines.

4. (10 points) R-9.7 p. 417

Draw the 11-entry hash table that results from using the has function, $h(k) = (3k + 5) \bmod 11$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.

0	1	2	3	4	5	6	7	8	9	10

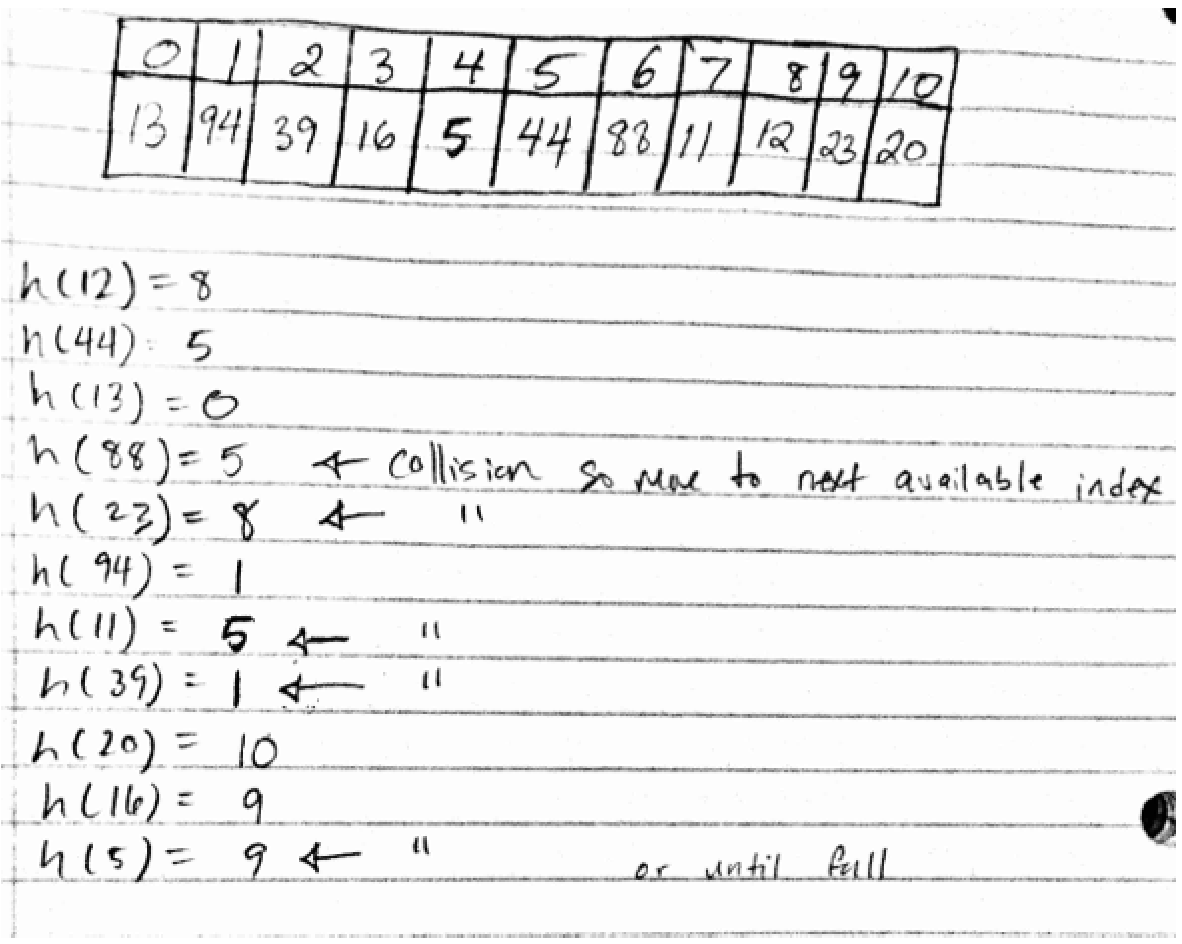
h(12) = (3(12)+5) % 11 = 41 % 11 = 3 r 8 = 8
h(44) = (3(44)+5) % 11 = 137 % 11 = 12 r 5 = 5
h(13) = (3(13)+5) % 11 = 44 % 11 = 4 r 0 = 0
h(88) = (3(88)+5) % 11 = 269 % 11 = 24 r 5 = 5
h(23) = (3(23)+5) % 11 = 74 % 11 = 6 r 8 = 8
h(94) = (3(94)+5) % 11 = 287 % 11 = 26 r 1 = 1
h(11) = (3(11)+5) % 11 = 38 % 11 = 3 r 5 = 5
h(39) = (3(39)+5) % 11 = 122 % 11 = 11 r 1 = 1
h(20) = (3(20)+5) % 11 = 65 % 11 = 5 r 10 = 10
h(16) = (3(16)+5) % 11 = 53 % 11 = 4 r 9 = 9
h(5) = (3(5)+5) % 11 = 20 % 11 = 1 r 9 = 9



(a)

5. (10 points) R-9.8 p. 417

What is the result of the previous exercise, assuming collisions are handled by linear probing?

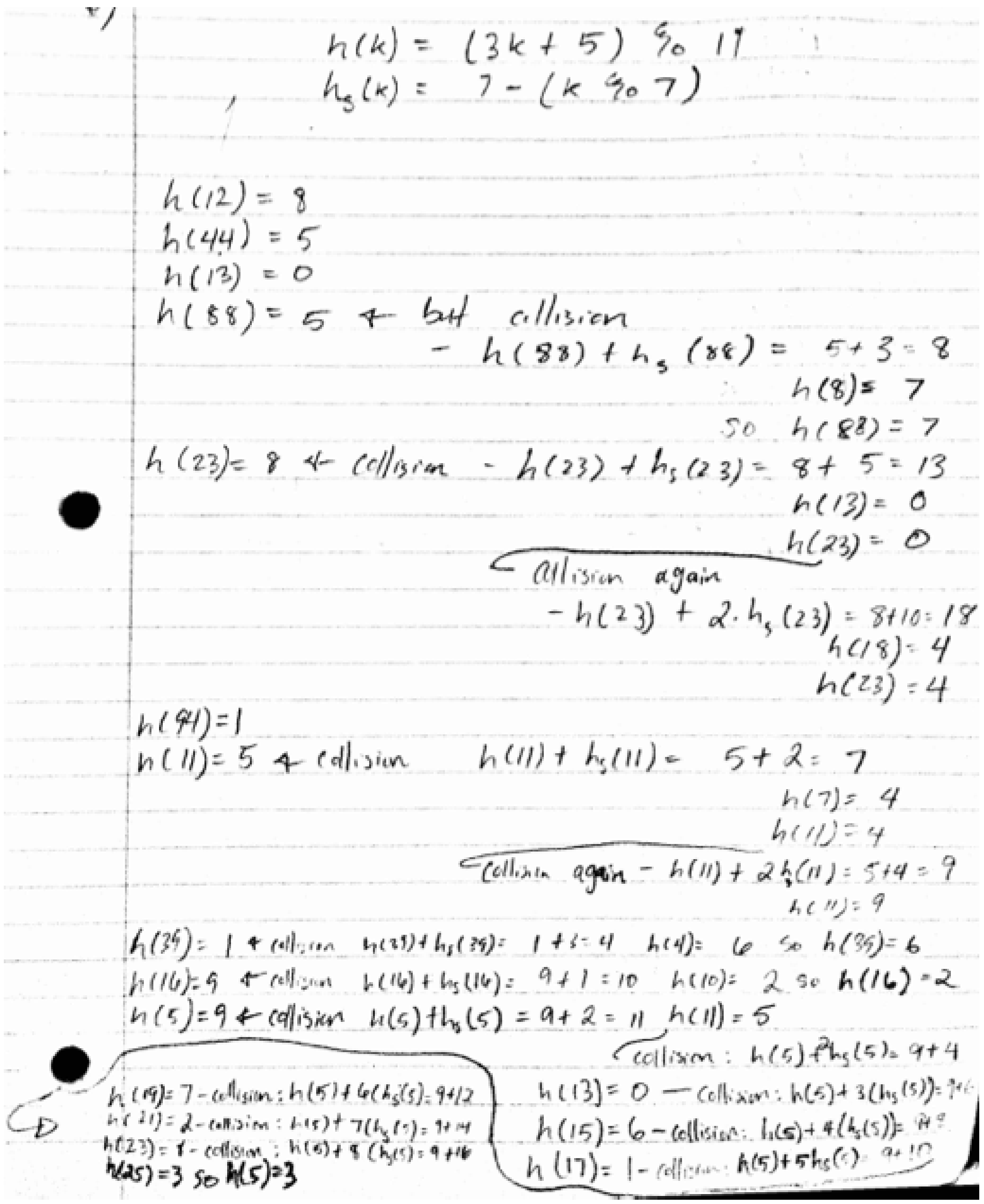
(a) 

0	1	2	3	4	5	6	7	8	9	10
13	94	39	16	5	44	88	11	12	23	20

$h(12) = 8$
$h(44)$: 5
$h(13) = 0$
$h(88) = 5$ $\leftarrow$ collision so move to next available index
$h(23) = 8$ $\leftarrow$ "
$h(94) = 1$
$h(11) = 5$ $\leftarrow$ "
$h(39) = 1$ $\leftarrow$ "
$h(20) = 10$
$h(16) = 9$
$h(5) = 9$ $\leftarrow$ " or until full

6. (10 points) R-9.10 p. 417

What is the result of Exercise R-9.7, when collisions are handled by double hashing using the secondary hash function $h_s(k) = 7 - (k \bmod 7)$?



$h(k) = (3k+5) \% 11$

$h_s(k) = 7 - (k \% 7)$

$h(12) = 8$

$h(44) = 5$

$h(13) = 0$

$h(88) = 5$ ← but collision

− $h(88) + h_s(88) = 5+3 = 8$

$h(8) = 7$

so $h(88) = 7$

$h(23) = 8$ ← collision − $h(23) + h_s(23) = 8+5 = 13$

$h(13) = 0$

$h(23) = 0$

collision again

− $h(23) + 2 \cdot h_s(23) = 8+10 = 18$

$h(18) = 4$

$h(23) = 4$

$h(94) = 1$

$h(11) = 5$ ← collision $h(11) + h_s(11) = 5+2 = 7$

$h(7) = 4$

$h(11) = 4$

collision again − $h(11) + 2h_s(11) = 5+4 = 9$

$h(11) = 9$

$h(39) = 1$ ← collision $h(39) + h_s(39) = 1+3 = 4$ $h(4) = 6$ so $h(39) = 6$

$h(16) = 9$ ← collision $h(16) + h_s(16) = 9+1 = 10$ $h(10) = 2$ so $h(16) = 2$

$h(5) = 9$ ← collision $h(5) + h_s(5) = 9+2 = 11$ $h(11) = 5$

collision : $h(5) + 2h_s(5) = 9+4$

$h(13) = 0$ — collision : $h(5) + 3(h_s(5)) = 9+6$

$h(15) = 6$ − collision : $h(5) + 4(h_s(5)) = 9+8$

$h(17) = 1$ − collision : $h(5) + 5h_s(5) = 9+10$

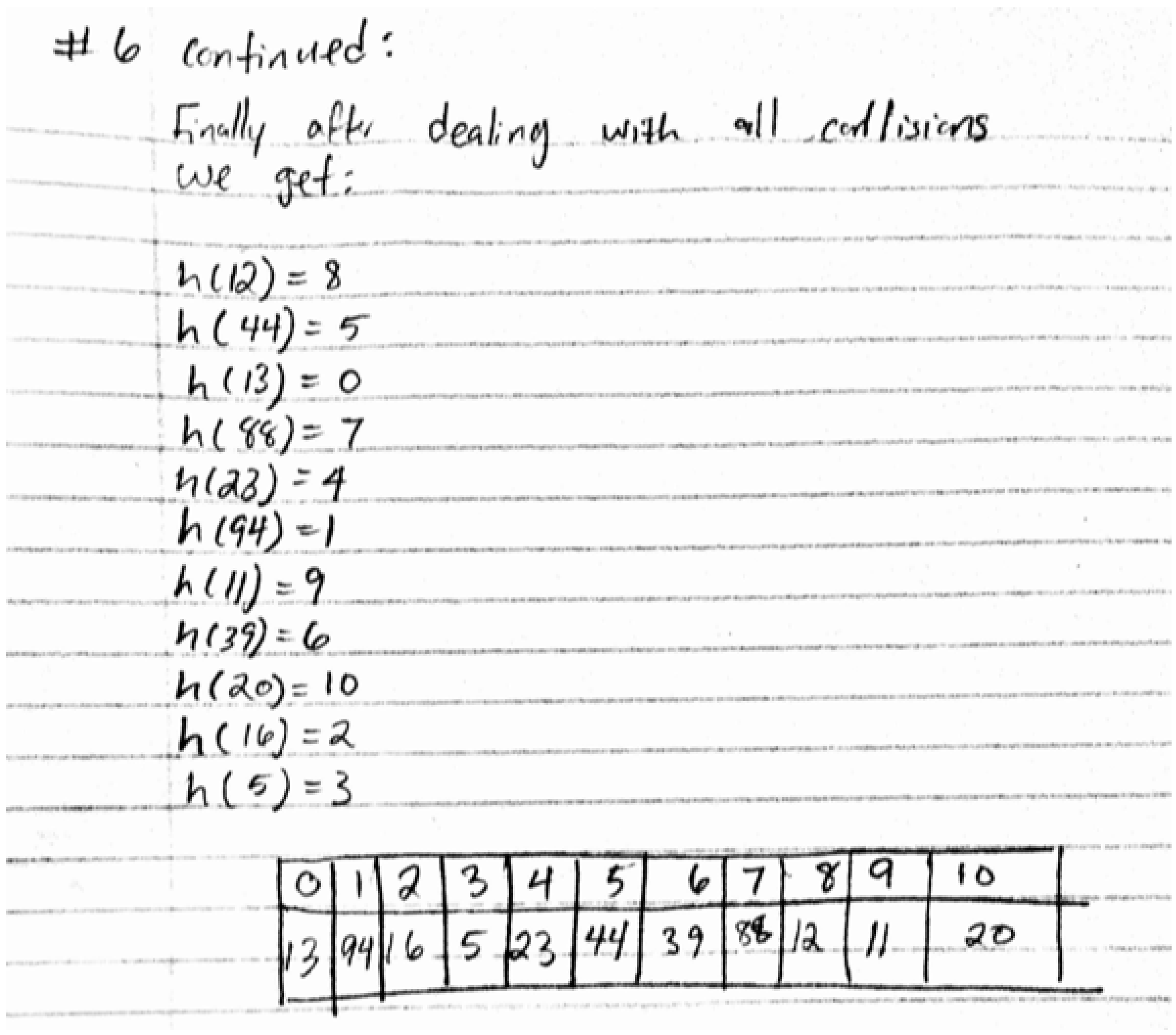
$h(19) = 7$ − collision : $h(5) + 6(h_s(5)) = 9+12$

$h(21) = 2$ − collision : $h(5) + 7(h_s(5)) = 9+14$

$h(23) = 1$ − collision : $h(5) + 8(h_s(5)) = 9+16$

$h(25) = 3$ so $h(5) = 3$

(a)



#6 continued:

Finally after dealing with all collisions we get:

$h(12) = 8$
$h(44) = 5$
$h(13) = 0$
$h(88) = 7$
$h(23) = 4$
$h(94) = 1$
$h(11) = 9$
$h(39) = 6$
$h(20) = 10$
$h(16) = 2$
$h(5) = 3$

0	1	2	3	4	5	6	7	8	9	10
13	94	16	5	23	44	39	88	12	11	20

(b)

7. (10 points) R-8.2 p. 361

   How long would it take to remove $\lceil log\, n \rceil$ smallest elements from a heap that contains n entries using the `removeMin()` operation?

   (a) Since each remove operation takes O(logn) the logn would there fore run in O(log^{2}n).

8. (10 points) R-8.7 p. 361

   An airport is developing a computer simulation of air-traffic control that handles events such as landings and takeoffs. Each event has a *time-stamp* that denotes the time when the event occurs. The simulation program needs to efficiently perform the following two fundamental operations:

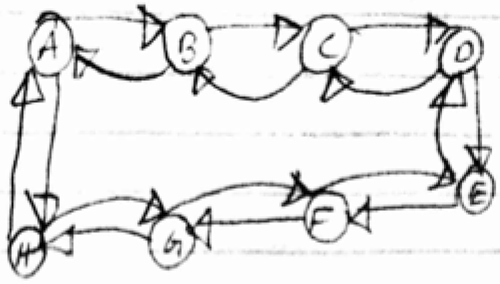
   - Insert an event with a given time-stamp (that is, add a future event)
   - Extract the event with smallest time-stamp (that is, determine the next event to process)

   Which data structure should be used for the above operations? Why?

   (a) Binary heap because we can extract the smallest element which would be at the top O(1). To insert elements it would also take O(logn). The keys would be the time stamp.

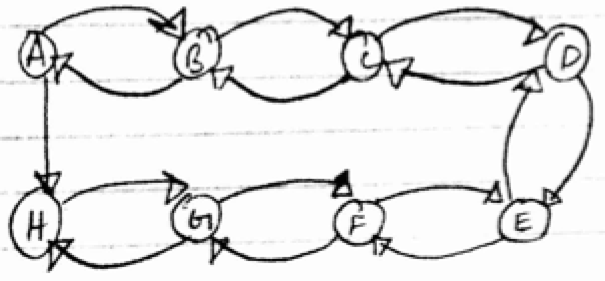
9. (10 points) R-13-3 and R-13-4, p. 654

R 13.3)



Euler path: A → B → C → D → E → F → G → H → A →
G → F → E → D → C → B → A

R 13.4)



Euler path: A → H → G → F → E → D → C → B → A →
C → D → E → F → G → H

(a)

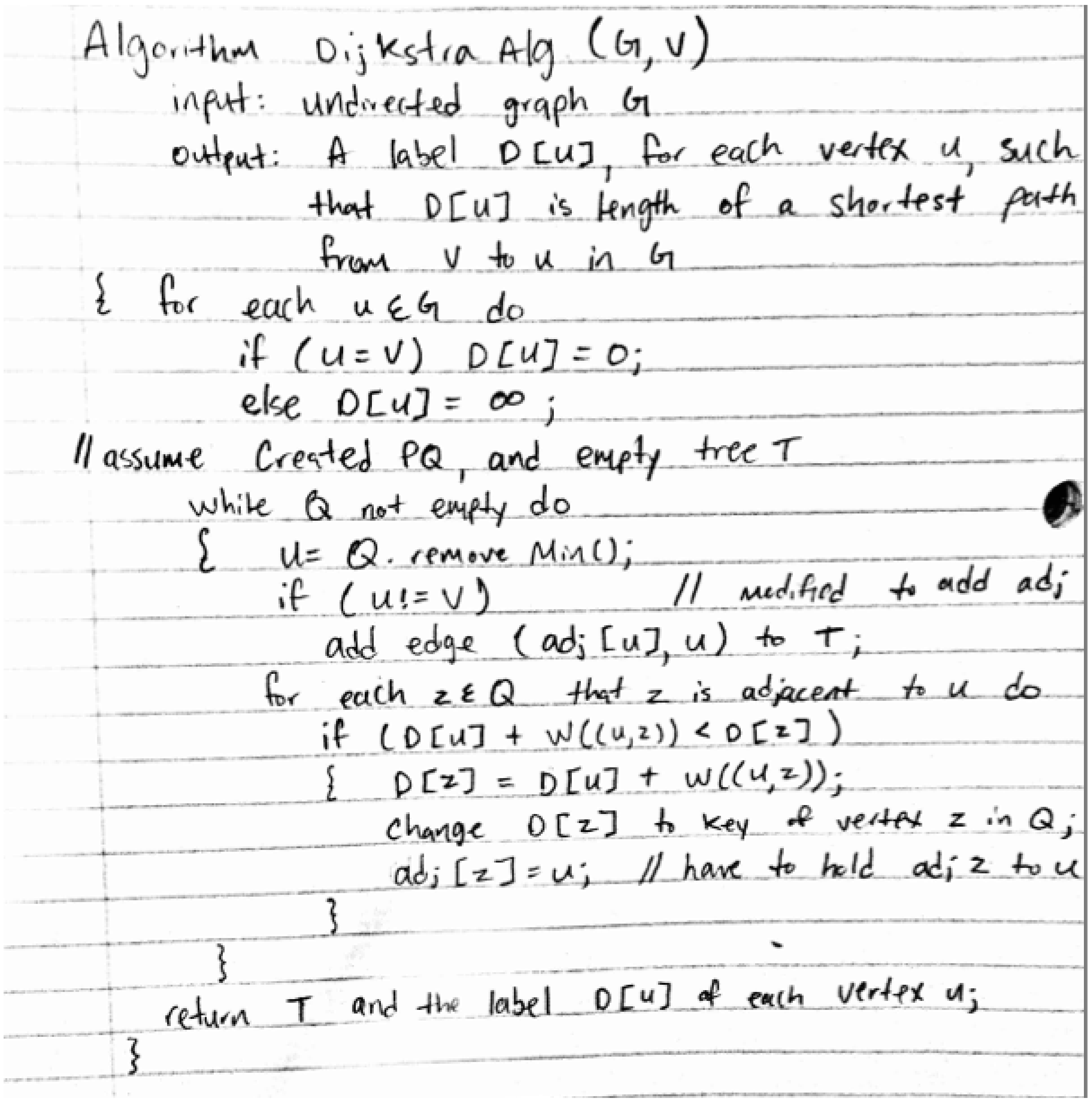
10. (10 points) R-13.8, p. 655

(a) Adjacency list since it is known to be better with fewer edges and can iterate in O(e+v) where e denotes edges and v is vertices.

(b) Adjacency matrix because it works better for dense graphs, and can iterate through all edges in O(v^2) time.

(c) Adjacency matrix since we can check adjacent in O(1) time.

11. (10 points) R-13.16, p. 656

(a) 

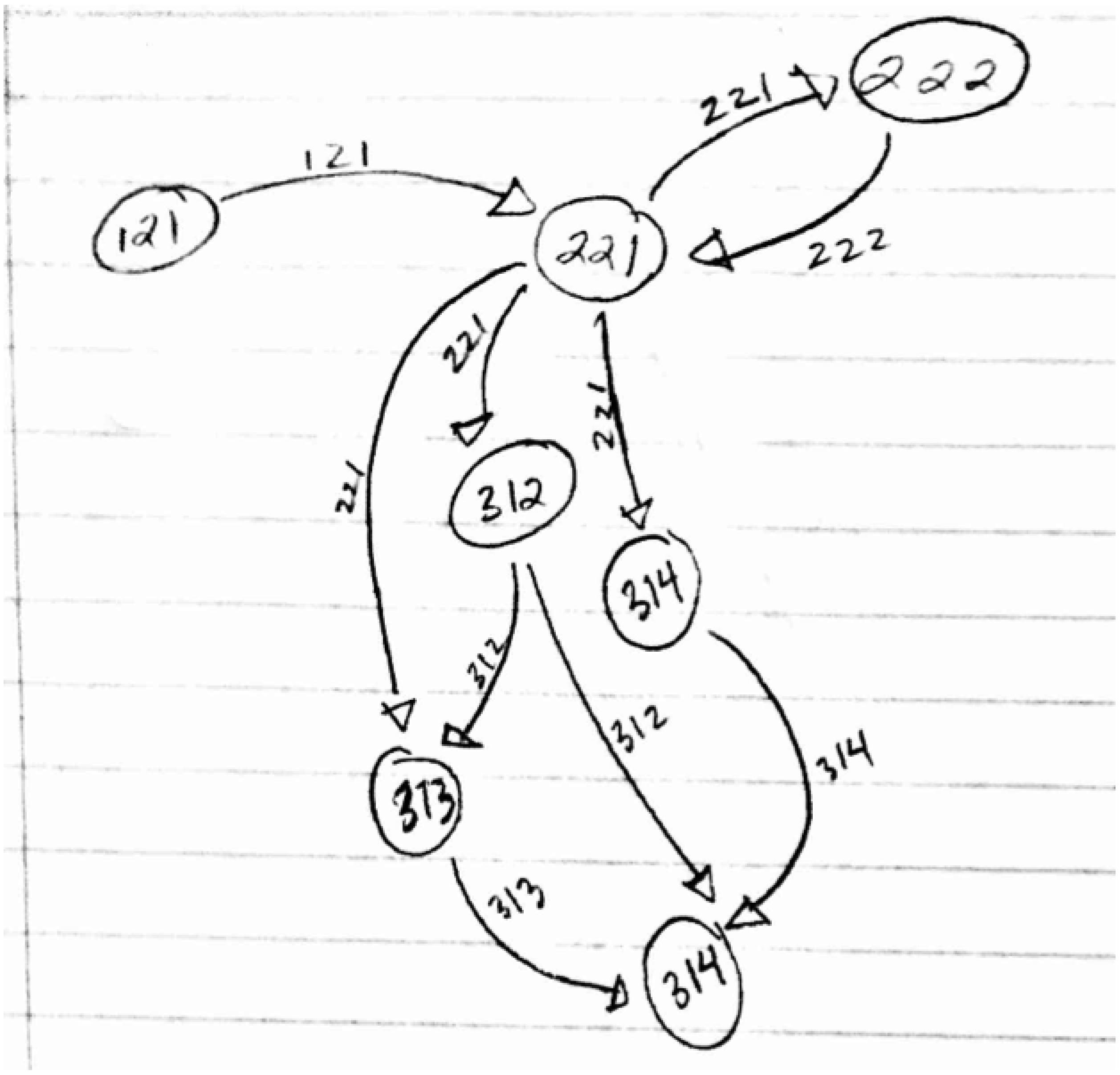
```
Algorithm DijkstraAlg (G, V)
    input: undirected graph G
    output: A label D[u], for each vertex u, such
            that D[u] is length of a shortest path
            from V to u in G
{ for each u∈G do
        if (u = V) D[u] = 0;
        else D[u] = ∞ ;
// assume Created PQ, and empty tree T
    while Q not empty do
    {   u= Q.removeMin();
        if (u!= V)                // modified to add adj
            add edge (adj[u], u) to T;
        for each z∈Q that z is adjacent to u do
            if (D[u] + w((u,z)) < D[z])
            {   D[z] = D[u] + w((u,z));
                change D[z] to key of vertex z in Q;
                adj[z] = u;  // have to hold adj z to u
            }
    }
    return T and the label D[u] of each vertex u;
}
```

12. (10 points) R-13.17, p. 656

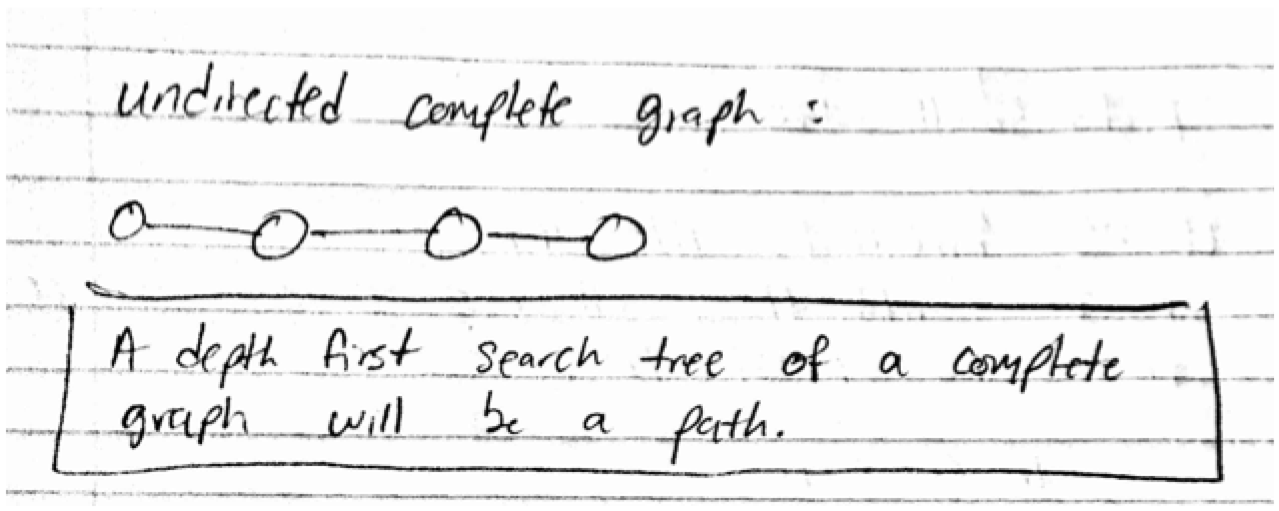
(a) Using Kruskal Algorithm we can build the following bridges to minimize the total construction cost:

    i. 3 to 5 (115)
    ii. 1 to 8 (120)
    iii. 2 to 8 (155)
    iv. 4 to 5 (160)
    v. 5 to 8 (170)
    vi. 6 to 7 (175)
    vii. 2 to 6 (180)

13. (10 points) You want to help CS/CSE freshman students to prepare their course schedules for the first two years in the lower level division. By building a directed graph suggest order in which they should schedule their courses taking into account their corresponding prerequisites. A set of vertices represents courses and a set of edges represents a dependence of a given course on a course prerequisite.

(a) 

14. (10 points) R-13.31, p. 656



undirected complete graph :

A depth first search tree of a complete graph will be a path.

(a)

15. (10 points) Write what the running time, and provide its justification, of the Dijkstra's algorithm is for a sparse and dense graph and the priority queue implemented based on

(a) a binary heap

i. Sparse: Creating the binary heap would take O(Vlog(V)), RemoveMin would be the same, decreaseKey would be O(Elog(V)) so therefore the total running time would be O((V+E)log(V)).

ii. Dense: Creating the binary heap, and RemoveMin would both take O(Vlog(V)) and decreaseKey would be O(V^2 log(V)), and therefore the total running time would be O(V^2 log(V)).

(b) an unsorted list

i. Sparse: Creating the binary heap would be O(V), RemoveMin would be O(V^2), decreaseKey would be O(E), so the total running time would be O(V^2+ E).

ii. Dense: Creating the binary heap would be O(V), RemoveMin, and decreaseKey would be O(V^2), the total running time would be O(V^2).

(c) a sorted list

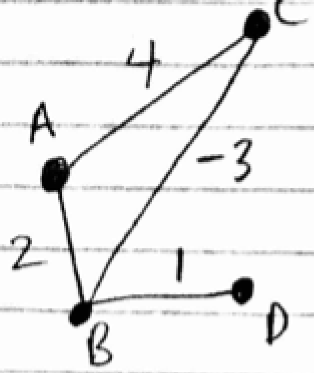
i. Sparse: Creating the binary heap would be O(V^2), RemoveMin would be O(V), decreaseKey would be O(E), and therefore the total running time was O(V^2+E).

ii. Dense: Creating the binary heap and decreaseKey would be O(V^2), RemoveMin would be O(V), therefore the total running time would be O(V^2).

16. (10 points) C-13.10, p. 658

(a) We can use a DFS type algorithm to traverse the graph. With modification we must not close circuits before all the edges are traversed. When there is a distinguished vertex which has an odd number of untraversed edges adjacent to it we can do a DFS traversal to close this circuit. If we do close and not all edges have been traversed we can then back track and find the last vertex we saw that had untraversed edges and make it a distinguished vertex. Then you try to find another edge that hasnt been traversed. When you do just attach the old circuit to a new circuit. This algorithm would be O(n+m) because it traverses like a DFS, and would spend at most O(m) time reconnecting edges.

17. (10 points) C-13.15, p. 659



Shortest path from A to

$= 3$

Relaxing (C, B) gives us $4-3=1$

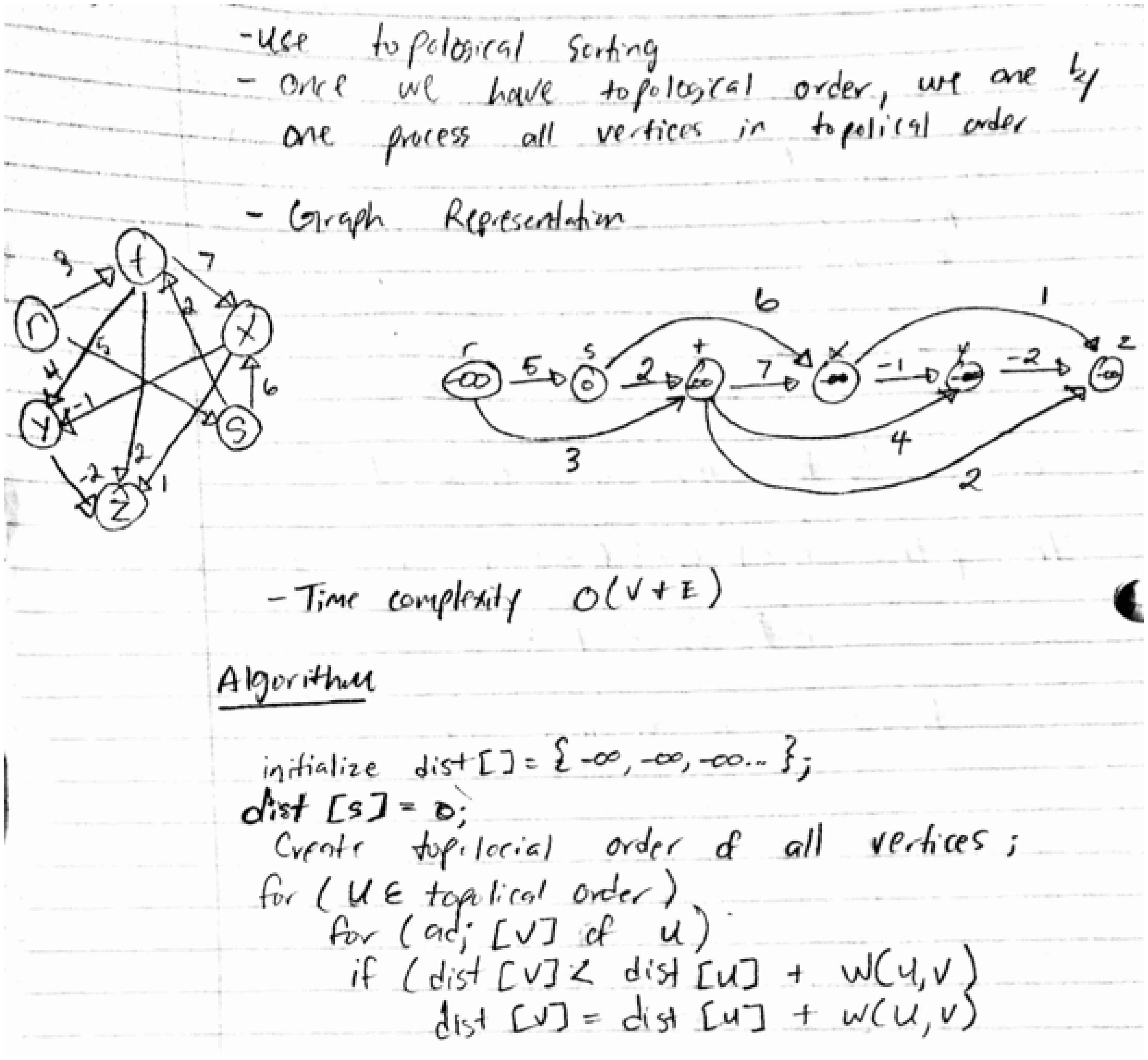
The path $A \rightarrow C \rightarrow B \rightarrow D$ Now gives us

$$4+(-3)+1=2$$

Hence it is incorrect since $2 \neq 3$

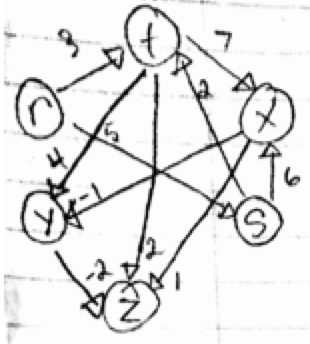
(a)

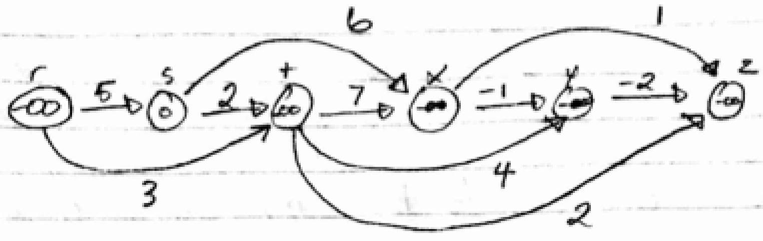
18. (10 points) C-13.18, p. 659



- Use topological Sorting
- Once we have topological order, we are b/ one process all vertices in topolical order

- Graph Representation





- Time complexity O(V + E)

Algorithm

```
initialize dist[] = {-∞, -∞, -∞..};
dist [s] = 0;
 Create topolocial order of all vertices ;
for (U ∈ topolical order)
    for (adj [V] of U)
      if (dist [V] < dist [U] + W(U,V)
           dist [V] = dist [U] + W(U,V)
```

(a)