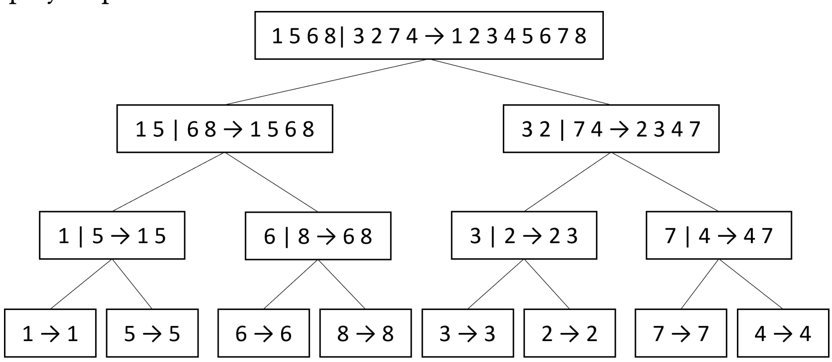
CSCI2100D 2023-24: Solution of Assignment 3[*]

- **Q1. [12 marks]** Let $A[0 \cdots 7] = [1, 5, 6, 8, 3, 2, 7, 4]$.
  - **(i). [6 marks]** Show the process of **mergesort**(A, 0, 7) to sort A in ascending order step by step.



  - **(ii). [6 marks]** Assume that we call **quicksort**(A, 0, 7) to sort A in ascending order, and the pivot position we randomly choose is 1. Show how the partition works and indicate the value of **nextsmallpos** during the invocation of **partition**(A, 0, 7, 1) step by step.

    Since the pivot position is 1, we have **pivotVal** = 5.

Step 1: j = 0, nextSmallpos = 0.	1	4	6	8	3	2	7	5
Step 2: j = 1, nextSmallpos = 1.	1	4	6	8	3	2	7	5
Step 3: j = 2, nextSmallpos = 2.	1	4	6	8	3	2	7	5
Step 4: j = 3, nextSmallpos = 2.	1	4	6	8	3	2	7	5
Step 5: j = 4, nextSmallpos = 3.	1	4	3	8	6	2	7	5
Step 6: j = 5, nextSmallpos = 4.	1	4	3	2	6	8	7	5
Step 7: j = 6, nextSmallpos = 4.	1	4	3	2	6	8	7	5

    Then we swap $A[4]$ and $A[7]$. It yields $A = [1, 4, 3, 2, 5, 8, 7, 6]$.

- **Q2. [12 marks]** Sort array $A[1 \cdots 7] = [9, 1, 10, 3, 2, 8, 4]$ in decreasing order by heap sort. (you may just show the array representation at each step.)
  - **(i). [6 marks]** Show the contents of A in the heap adjust process to make it a min-heap step by step.

Step 1: Adjust node with id 3.	9	1	4	3	2	8	10

[*]Departmental Guideline for Plagiarism (Department of Systems Engineering and Engineering Management): If a student is found plagiarizing, his/her case will be reported to the Department Examination Panel. If the case is proven after deliberation, the student will automatically fail the course in which he/she committed plagiarism. The definition of plagiarism includes copying of the whole or parts of written assignments, programming exercises, reports, quiz papers, mid-term examinations and final examinations. The penalty will apply to both the one who copies the work and the one whose work is being copied, unless the latter can prove his/her work has been copied unwittingly. Furthermore, inclusion of others' works or results without citation in assignments and reports is also regarded as plagiarism with similar penalty to the offender. A student caught plagiarizing during tests or examinations will be reported to the Faculty office and appropriate disciplinary authorities for further action, in addition to failing the course.

Step 2: Check node with id 2. No violation

Step 3: Adjust node with id 1.

1	9	4	3	2	8	10

Step 4: Adjust node with id 2.

1	2	4	3	9	8	10

– **(ii). [6 marks]** Using the min-heap of Part (i), show the contents of A in the sorting process of swaping elements in the array step by step.

Step 1: Swap node id 7 and node id 1; reduce heap size by 1.

10	2	4	3	9	8	**1**

Step 2: Adjust.

2	3	4	10	9	8	**1**

Step 3: Swap node id 6 and node id 1; reduce heap size by 1.

8	3	4	10	9	**2**	**1**

Step 4: Adjust.

3	8	4	10	9	**2**	**1**

Step 5: Swap node id 5 and node id 1; reduce heap size by 1.

9	8	4	10	**3**	**2**	**1**

Step 6: Adjust.

4	8	9	10	**3**	**2**	**1**

Step 7: Swap node id 4 and node id 1; reduce heap size by 1.

10	8	9	**4**	**3**	**2**	**1**

Step 8: Adjust.

8	10	9	**4**	**3**	**2**	**1**

Step 9: Swap node id 3 and node id 1; reduce heap size by 1.

9	10	**8**	**4**	**3**	**2**	**1**

Step 10: No violation.

Step 11: Swap node id 2 and node id 1; reduce heap size by 1.

10	**9**	**8**	**4**	**3**	**2**	**1**

Step 12: No violation.

Step 13: The heap size now is 1. Sort finishes.

10	**9**	**8**	**4**	**3**	**2**	**1**

- **Q3. [14 marks]** Assume that we have a hash table with size $m = 13$ and the hash function $h(k) = 2k\%13$. We use linear probing to address collisions. Answer the following questions.

  – **(i). [7 marks]** Given an empty hash table, show the hash table when inserting 10, 2, 23, 13, 1, 9, 17 in order step by step. Inserting 10:

							10					

Inserting 2:

				2			10					

Inserting 23:

				2			10	23				

Inserting 13:

13				2			10	23				

Inserting 1:

13		1		2			10	23				

Inserting 9:

13		1		2	9		10	23				

Inserting 17:

13		1		2	9		10	23	17			

  – **(ii). [7 marks]** Given the following hash table, show the records examined when searching for 42.

0				4				8				12
0					22	16	10	30	24	42	25	

When searching for 42, the records 16,10,30,24,42 are examined in order.

- **Q4. [14 marks]** Assume that we have a hash table with size $m = 13$ and we use double hashing to address collisions. The double hashing function is $h(k, i) = (h(k)+i\cdot h'(k))\%m$, where $h(k) = k\%13$ and $h'(k) = 1 + k\%3$. Answer the following questions.
  - **(i). [7 marks]** Given an empty hash table, show the hash table when inserting 14, 2, 18, 36, 31, 23, 42 in order step by step.

Inserting 14: $h(14, 0) = 1$,

	14											

Inserting 3: $h(2, 0) = 2$,

	14	2										

Inserting 18: $h(18, 0) = 5$

	14	2			18							

Inserting 36: $h(36, 0) = 10$,

	14	2			18					36		

Inserting 31: $h(31, 0) = 5$, $h(31, 1) = 7$

	14	2			18		31			36		

Inserting 23: $h(23, 0) = 10$, $h(23, 1) = 0$,

23	14	2			18		31			36		

Inserting 42: $h(42, 0) = 3$

23	14	2	42		18		31			36		

  - **(ii). [7 marks]** Given the following hash table, show the records examined when searching for 44.

0				4				8				12
		15		30	5			21		10	24	38

When searching for 44, since $h(44, 0) = 5$, $h(44, 1) = 8$, and $h(44, 2) = 11$, $h(44, 3) = 1$, the records 5, 21, 24 are examined in order

- **Q5. [38 marks]** Consider the directed graph G_1 as shown in Fig. 1. Answer the following questions.

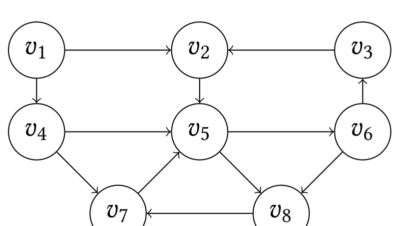


Fig. 1. Directed Graph G_1 for Q5

  - **(i). [2 mark]** Calculate the out-degree and the in-degree of v_5.
    The out-degree of v_5 is 2. The in-degree of v_5 is 3.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
minlength	0	1	4	1	2	3	2	3
prev	*nil*	v_1	v_6	v_1	v_2	v_5	v_4	v_5

Table 1. **minlength** and **prev** Array of Q5(v)

- **(ii). [2 mark]** Whether the path $v_1 \longrightarrow v_4 \longrightarrow v_5 \longrightarrow v_6 \longrightarrow v_8 \longrightarrow v_7$ is a simple path? Justify your answer.
  Yes, because all the edges exist and all the nodes are distinct.
- **(iii). [8 marks]** For G_1, show both its adjacency list representation and its adjacency matrix representation. (The nodes should be in ascending order of ID.)

$v_1 \rightarrow head \leftrightarrow v_2 \leftrightarrow v_4 \leftrightarrow tail$
$v_2 \rightarrow head \leftrightarrow v_5 \leftrightarrow tail$
$v_3 \rightarrow head \leftrightarrow v_2 \leftrightarrow tail$
$v_4 \rightarrow head \leftrightarrow v_5 \leftrightarrow v_7 \leftrightarrow tail$
$v_5 \rightarrow head \leftrightarrow v_6 \leftrightarrow v_8 \leftrightarrow tail$
$v_6 \rightarrow head \leftrightarrow v_3 \leftrightarrow v_8 \leftrightarrow tail$
$v_7 \rightarrow head \leftrightarrow v_5 \leftrightarrow tail$
$v_8 \rightarrow head \leftrightarrow v_7 \leftrightarrow tail$

Adjacency List for Q5(iii)

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1	0	1	0	1	0	0	0	0
v_2	0	0	0	0	1	0	0	0
v_3	0	1	0	0	0	0	0	0
v_4	0	0	0	0	1	0	1	0
v_5	0	0	0	0	0	1	0	1
v_6	0	0	1	0	0	0	0	1
v_7	0	0	0	0	1	0	0	0
v_8	0	0	0	0	0	0	1	0

Adjacency Matrix for Q5(iii)

Fig. 2. Solution of Q5(iii)

- **(iv). [8 marks]** Traverse G_1 using breadth-first search with v_1 as the source, assuming that the out-neighbors of a node are visited in ascending order of ID. Show the process and the contents of queue Q step by step. You may use 0 to denote the color to be white, 1 to denote the color to be gray, and 2 to denote the color to be black.
  The answer can be found in Figure 3.
- **(v). [8 marks]** According to the results of Part (iv), show the contents of **minlength** array and **prev** array respectively.
  The answer can be found in Table 1.
- **(vi). [4 marks]** Show how to get the minimum length path from the source v_1 to v_7 using the **minlength** array and **prev** array. Justify your answer.
  Get the previous node of v_7 in prev array, which is v_4; get the previous node of v_4, which is v_1, i.e., the source. We get the path: $v_1 \longrightarrow v_4 \longrightarrow v_7$, whose length is 2.
- **(vii). [6 marks]** Draw the BFS tree.
  The answer can be found in Figure 4.

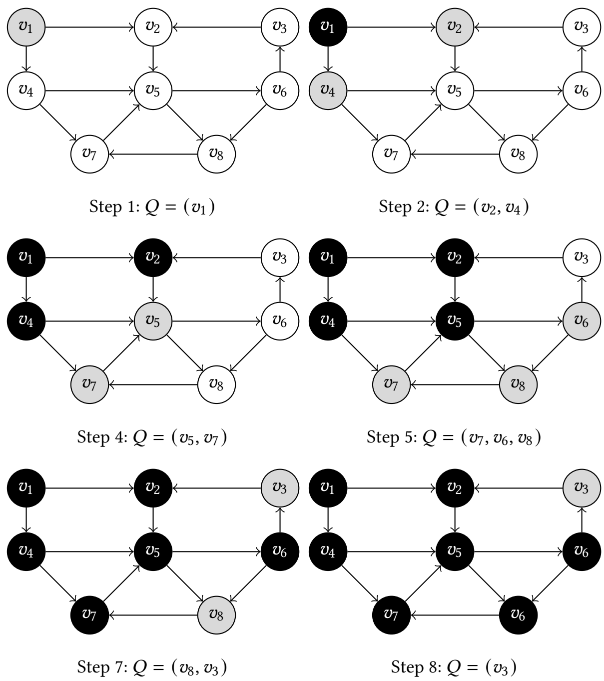


Fig. 3. Solution of Q5(iv)

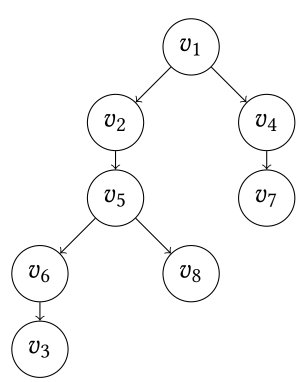


Fig. 4. Solution of Q5(vii)

- **Q6. [10 marks]** Given an undirected graph $G = (V, E)$, A triangle consists of three nodes in G that are pairwise adjacent. More formally, a triangle in G is triplet (u, v, w) where $u, v, w \in V$ and $(u, v), (v, w), (w, u) \in E$. Consider the undirected graph G_2 as shown in Fig. 5. There are two triangles in G_2, (v_0, v_1, v_2) and (v_1, v_2, v_3).

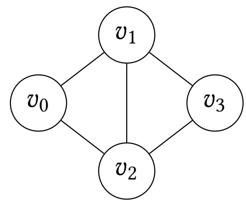


Fig. 5. An Undirected Graph G_2 for Q6

- **(i). [4 marks]** Please design an algorithm TriangleCounting(G) using pseudo-code with the provided Graph ADT, to count the number of triangles in G.

  Graph **ADT**.
  * **`Vertices(G)`:** Lists all vertices u in G.
  * **`Neighbors(G,u)`:** Lists all vertices v such that there is an edge between the vertex u and the vertex v.
  * **`Adjacent(G,u,v)`:** Tests whether there is an edge between the vertex u and the vertex v.

  The algorithm can be designed using iterative approach as follows:

  Algorithm `TriangleCounting`(G)
  $count \leftarrow 0$
  for $u \in Vertices(G)$
      for $v \in Neighbors(G, u)$
          for $w \in Neighbors(G, v)$
              if $Adjacent(G, u, w)$
                  $count \leftarrow count + 1$
  return $count/6$

- **(ii). [2 marks]** Assume that G has n nodes and m edges, and the degree of any node u in G is smaller than d_{max}. What is the time complexity of TriangleCounting(G) expressed in terms of n, m, and d_{max} if the graph ADT is implemented using an adjacency matrix? Justify your answer.
  Since the graph ADT is implemented using an adjacency matrix, the $Vertices(G)$ costs $O(n)$ time, the $Neighbors(G, u)$ costs $O(n)$ time, the $Neighbors(G, v)$ costs $O(n)$ time and the $Adjacent(G, u, v)$ costs $O(1)$ time. Therefore, the time complexity of the algorithm is $O(n \times n \times n \times 1) = O(n^3)$

- **(iii). [2 marks]** Assume that G has n nodes and m edges, and the degree of any node u in G is smaller than d_{max}. What is the time complexity of $TriangleCounting(G)$ expressed in terms of n, m, and d_{max} if the graph ADT is implemented using an adjacency list? Justify your answer.

Since the graph ADT is implemented using an adjacency list, the $Vertices(G)$ and $Neighbors(G, u)$ enumerate each edge exactly once, thus cost $O(m)$ time in total. Besides, the $Neighbors(G, u)$ costs $O(d_{max})$ and the $Adjacent(G, u, v)$ costs $O(d_{max})$ time. Therefore, the time complexity of the algorithm is $O(m \times d_{max} \times d_{max}) = O(md_{max}^2)$

- **(iv). [2 marks]** Assume that the graph ADT is implemented using an adjacency list, and the output of Neighbors(G, u) is guaranteed to be sorted in ascending order by node ID. Please design a more efficient TriangleCounting(G) using pseudo-code.

The algorithm can be designed using iterative approach as follows:

```
Algorithm TriangleCounting(G)
count ← 0
for u ∈ Vertices(G)
   for v ∈ Neighbors(G, u)
      N1 ← Neighbor(G, u)
      N2 ← Neighbor(G, v)
      w1 ← N1.head.next
      w2 ← N2.head.next
      while w1! = N1.tail and w2! = N2.tail
         if w1.element < w2.element
            w1 ← w1.next
         else if w1.element > w2.element
            w2 ← w2.next
         else
            count ← count + 1
            w1 ← w1.next
            w2 ← w2.next
return count/6
```

The time complexity is $O(m \times d_{max}) = O(md_{max})$