招生學年度	102	招生類別	轉學考招生考試
系所班別	資訊工程學系三年級		
科目名稱	資料結構		
注意事項	禁止使用掌上型計算機		

1. (25%) Given 10 distinct identifiers GA, A, G, L, A2, A1, A3, A4 and E, please design a hashing function to explain the following terms: (a) hash table, (b) bucket, (c) slot, (d) collision, (e) overflow, and (f) loading density.

2. (15%) (a) What order makes a B-tree a 2-3-4 tree? (b) What is the number of keys which make all B-trees of order ***2*** full binary trees? (c) If we have to split a full node ***p*** while inserting a key. Let ***p*** be of the format $m, A_0, (E_1, A_1), \ldots, (E_n, A_n)$, and $E_i < E_{i+1}, 1 <= i < m$, then what about the format of two split nodes?

3. (15%) Figure 1 is an example of B^+-Tree, please draw the corresponding B^+-Trees after inserting elements with key 27, 14, and 86.

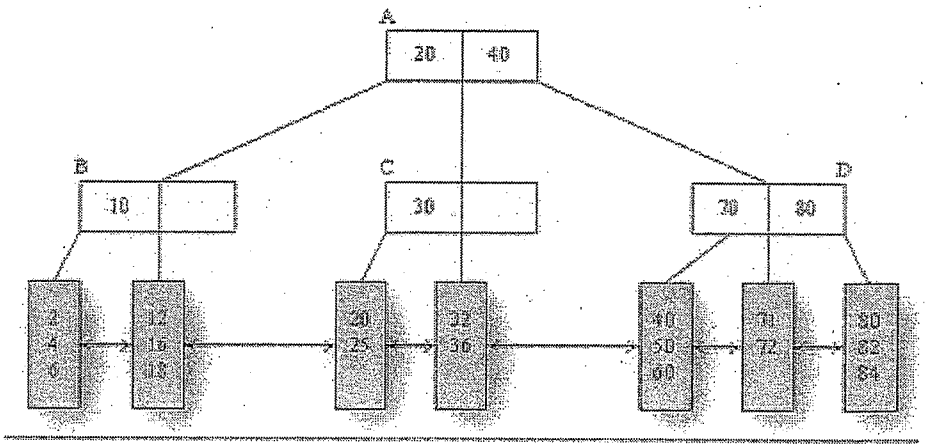

Figure 1 An example of B+-Tree

4. (10%) Please draw a 4-node complete graph and its all spanning trees?

5. (15%) Given the graph G of Figure 2. (a) Please complete the representation of its adjacency list. Hint: its partial adjacency list is shown in Figure 3 and be noticed that for vertices in each linked list must be in an ascending order. (b) If a ***XXXSearch()***, as can be seen in the following, is initiated from vertex 0, then write down the order of visited vertices.

招生學年度	102	招生類別	轉學考招生考試
系所班別	資訊工程學系三年級		
科目名稱	資料結構		
注意事項	禁止使用掌上型計算機		

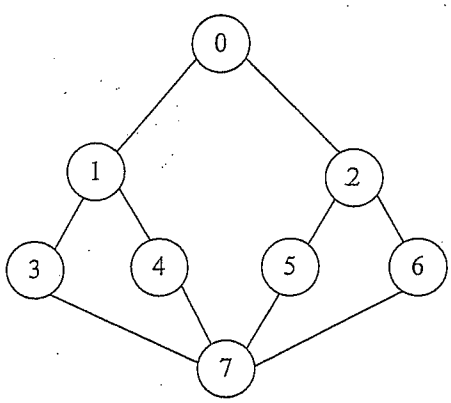

Figure 2 The input graph G

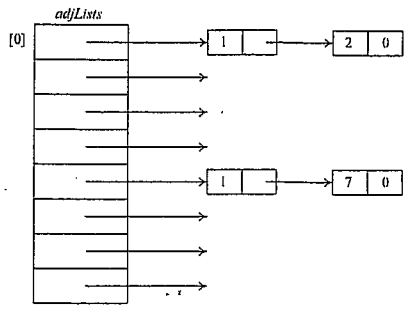

Figure 3 A partial adjacency list corresponding to the input graph G

```
virtual void Graph::XXXSearch()
{
   visited = new bool[n];
   fill (visited, visited + n, false);
   XXXSearch(0); // start search from vertex 0
   delete [] visited;
}
virtual void Graph:: XXXSearch(const int v)
{
   visited[v] = true;
   for ( each vertex w adjacent to v)
      if (!visited[w]) XXXSearch(w);
}
```

6. (20%) Please summarize the comparison of sorting methods by filling the following table.

Method	Time complexity		Extra Space	Stable	Best applicable cases
	Worst	Average			
Selection sort	(1)	(6)	O(1)	(15)	-
Insertion sort	(2)	(7)	(11)	(16)	Small n and few records are LOO
Quick sort	(3)	(8)	(12)	(17)	(20)
Merge sort	(4)	(9)	(13)	(18)	-
Heap sort	(5)	(10)	(14)	(19)	-