1. Implement operation UNION for sorted singly-linked list representations of sets. Give the order of magnitude of its running time when applied to sets of sizes m, n.

2. We wish to improve the speed of operations of an open hash table with \(B_1\) buckets by replacing it with another open hash table with \(B_2\) buckets. Write a procedure constructing the new hash table from the old one.

3. Let \(n\) be any positive integer. Is it possible to construct a priority queue \(A\) of size \(n\) with the property that at every iteration of operation DELETEMIN the process substituting the process with the smallest priority will be eventually placed on the last position? Provide either a construction or an argument showing that a priority queue with the above property does not exist.

```plaintext
function DELETEMIN(var A: PRIORITYQUEUE): ↑processtype;
var
  i, j: integer;
  temp: processtype;
  minimum: ↑processtype;
begin
  new(minimum);
  minimum ↑ := A.contents[1];
  A.contents[1] := A.contents[A.last];
  A.last := A.last – 1;
  i := 1;
  while i <= A.last div 2 do begin
    if (2*i = A.last) or (p(A.contents[2*i]) < p(A.contents[2*i + 1])) then
      j := 2*i
    else
      j := 2*i + 1
    if p(A.contents[i]) > p(A.contents[j]) then begin
      temp := A.contents[i];
      A.contents[i] := A.contents[j];
      A.contents[j] := temp;
      i := j
    end
  end;
  return (minimum)
end;
```
4. Find a formula for the total number of calls occurring during the insertion of \( n \) elements into an initially empty set. Assume that the insertion process fills up the binary search tree level-by-level. Leave your answer in the form of a sum.

\[
\text{procedure INSERT}(x: \text{elementtype}; \text{var A: SET});
\begin{align*}
&\text{beg} \begin{align*}
&\text{if A = nil then begin} \\
&\quad \text{new(A);} \\
&\quad A.K.element := x; \\
&\quad A.K.leftchild := \text{nil;} \\
&\quad A.K.rightchild := \text{nil} \\
&\text{end} \\
&\text{else if x < A.K.element then} \\
&\quad \text{INSERT(x, A.K.leftchild)} \\
&\text{else if x > A.K.element then} \\
&\quad \text{INSERT(x, A.K.rightchild)} \\
&\text{end};
\end{align*}
\end{align*}
\]

5. DELETE operation is applied to a BST of height \( k \). What are the maximal and the minimal numbers of calls of DELETEMIN, which may occur during the execution? Justify your answer.

\[
\text{function DELETEMIN( var A: SET ): elementtype;}
\begin{align*}
&\text{var} \\
&\quad \text{minimum: integer;} \\
&\text{begin} \\
&\quad \text{if A.K.leftchild = nil then begin} \\
&\quad \quad \text{minimum := A.K.element;} \\
&\quad \quad A := A.K.rightchild; \\
&\quad \quad \text{return (minimum)} \\
&\quad \text{end} \\
&\quad \text{else begin} \\
&\quad \quad \text{minimum := DELETEMIN(A.K.leftchild);} \\
&\quad \quad \text{return (minimum)} \\
&\quad \text{end} \\
&\text{end}
\end{align*}
\]
procedure DELETE(x: elementtype; var A: SET);
  begin
    if A <> nil then
      if x < A↑.element then
        DELETE(x, A↑.leftchild)
      else if x > A↑.element then
        DELETE(x, A↑.rightchild)
      else if (A↑.leftchild = nil) and (A↑.rightchild = nil) then
        A := nil
      else if A↑.leftchild = nil then
        A := A↑.rightchild
      else if A↑.rightchild = nil then
        A := A↑.leftchild
      else
        A↑.element := DELETEMIN(A↑.rightchild)
  end;

6. (i) Trace the execution of Dijkstra’s algorithm applied to the directed graph described by the following adjacency matrix. Include all values of arrays D and P at all stages of the execution. Describe all resulting shortest paths.


(ii) Provide an example of a directed graph with n vertices with the property that at every step of the execution of Dijkstra’s algorithm exactly one field of the array D is updated.

procedure Dijkstra;
  begin
    S := {1};
    for i := 2 to n do begin
      D[i] := C[1,i];
      P[i] := 1
    end;
    for i := 1 to n-1 do begin
      choose a vertex w in V-S such that
      D[w] is minimum;
      add w to S;
      for each vertex v in V-S do
        if D[v] > D[w] + C[w,v] then begin
          D[v] := D[w] + C[w,v];
          P[v] := w
        end
    end
  end;
7. Depth-first search dfs and breath-first search bfs graph traversal algorithms are applied to a full binary tree of height k. What are the maximal numbers of vertices occurring inside the stack of the dfs and inside the queue of the bfs during the execution? How many times are those maximal sizes repeated? Assume that the vertices are ordered level-by-level and that both algorithms respect this ordering.

**procedure** dfs(v: vertex);

```
var
  x,y: vertex;
  S: STACK of vertex;
begin
  mark[v] := visited;
  PUSH(v,S);
  while not EMPTY(S) do begin
    x := TOP(S):
    if there is an unvisited vertex y inside L[x] then begin
      mark[y] := visited;
      PUSH(y,S)
    end
    else
      POP(S)
  end
end;
```

**procedure** bfs(v: vertex);

```
var
  x,y: vertex;
  Q: QUEUE of vertex;
begin
  mark[v] := visited;
  ENQUEUE(v,Q);
  while not EMPTY(Q) do begin
    x := FRONT(Q):
    DEQUEUE(Q);
    for each vertex y inside L[x] do
      if mark[y] = unvisited then begin
        mark[y] := visited;
        ENQUEUE(y,Q)
      end
  end
end;
```