Solutions of the review problems

1. The solution presented below is a direct modification of operation INTERSECTION discussed in the textbook on pages 115-117.

```plaintext
procedure UNION ( ahead, bhead: ↑celltype; var pc: ↑celltype);
var
    acurrent, bcurrent, ccurrent: ↑celltype;
begin
    new(pc);
    acurrent := ahead ↑.next;
    bcurrent := bhead ↑.next;
    ccurrent := pc;
    while (acurrent <> nil) and (bcurrent <> nil) do begin
        if acurrent ↑.element = bcurrent ↑.element then begin
            new(ccurrent ↑.next);
            ccurrent := ccurrent ↑.next;
            ccurrent ↑.element := acurrent ↑.element;
            acurrent := acurrent ↑.next;
            bcurrent := bcurrent ↑.next
        end
        else if acurrent ↑.element < bcurrent ↑.element then begin
            new(ccurrent ↑.next);
            ccurrent := ccurrent ↑.next;
            ccurrent ↑.element := acurrent ↑.element;
            acurrent := acurrent ↑.next
        end
        else begin
            new(ccurrent ↑.next);
            ccurrent := ccurrent ↑.next;
            ccurrent ↑.element := bcurrent ↑.element;
            bcurrent := bcurrent ↑.next
        end
    end;
    if acurrent = nil then
        while bcurrent <> nil do begin
            new(ccurrent ↑.next);
            ccurrent := ccurrent ↑.next;
            ccurrent ↑.element := bcurrent ↑.element;
            bcurrent := bcurrent ↑.next
        end
    else
        while acurrent <> nil do begin
            new(ccurrent ↑.next);
            ccurrent := ccurrent ↑.next;
```
ccurrent↑.element := acurrent↑.element;
acurrent := acurrent↑.next
end;
ccurrent↑.next := nil
end; {UNION}

If m,n are the sizes of sets A,B, then the running time of operation UNION is O(m+n).

2. We follow the conventions of the implementation of DICTIONARY ADT presented on pages 124-125 of the textbook.

DICTIONARY1 = array[0..B1-1] of ↑celltype;
DICTIONARY2 = array[0..B2-1] of ↑celltype;

We assume that operations MAKENULL1, MAKENULL2, MEMBER1, MEMBER2, INSERT1, INSERT2, DELETE1, DELETE2 and hash functions h1 and h2 corresponding to sizes B1 and B2 are implemented inside data types DICTIONARY1, DICTIONARY2.

procedure COPY(A1: DICTIONARY1, var A2: DICTIONARY2);
begin
MAKENULL2(A2);
for i := 0 to B1-1 do begin
  current := A1[i];
  while current <> nil do begin
    INSERT2(current↑.element, A2);
    current := current↑.next
  end
end
end; {COPY}

3. The following algorithm shows how to attach numbers form 1 to n to the nodes of the tree corresponding to the array representing the priority queue.

attach number n to the node at position n;
i := n-1;
while i <> 0 do begin
  take the path starting at the root and ending at the node corresponding to the array index i;
pull all the nodes on this path up one level;
  attach number n-i to the node being removed from the tree;
i := i-1;
end;
4. Let $2^k \leq n < 2^{k+1}$. Then $k = \left\lfloor \log_2 n \right\rfloor$ and the number of calls equals
\[ \sum_{i=0}^{k-1} (i+1)2^i + (k+1)(n-2^k + 1). \]

5. The minimal number is 0 and the maximal number is $k$.

6. (i)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>initial</td>
<td>{1}</td>
<td>-</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>{1,3}</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>{1,3,2}</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>{1,3,2,4}</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{1,3,2,4,5}</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{1,3,2,4,5,6}</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>


(ii) The adjacency matrix:

\[
A[1,2]=1, \\
\]

7. dfs: maximal size $k+1$; it is repeated $2^k$ times.
   bfs: maximal size $2^k$; it occurs only once.