Algorithm Specification

Algorithm Specification

- Natural language like English: we should ensure that each & every statement is definite.

- Graphic representation called flowchart: This method will work well when the algorithm is small & simple.

- Pseudo-code: Pseudo code is compact and has a definite syntax of its own. Algorithms appear as programs, which resembles language like Pascal & C.

```
Algorithm Kruskal(E, cost, n, t)
// E is the set of edges in G. G has n vertices. cost[u, v] is the
// cost of edge (u, v). t is the set of edges in the minimum-cost
// spanning tree. The final cost is returned.
{
    Construct a heap out of the edge costs using Heapify;
    for i := 1 to n do parent[i] := -1;
    // Each vertex is in a different set.
    i := 0; mincost := 0.0;
    while ((i < n - 1)  and (heap not empty)) do
    {
        Delete a minimum cost edge (u, v) from the heap
        and reheapify using Adjust;
        j := Find(u); k := Find(v);
        if (j ≠ k) then
        {
            i := i + 1;
            t[i, 1] := u; t[i, 2] := v;
            mincost := mincost + cost[u, v];
            Union(j, k);
        }
    }
    if (i ≠ n - 1) then write ("No spanning tree");
    else return mincost;
}
```

Algorithm Specification

- Comments begin with // and continue until the end of line.
- Blocks represent compound statements. A compound statement is a sequence of one or more statements. Blocks are indicated with matching braces { and }.
- An identifier begins with a letter. The data types of variables are not explicitly declared.

Algorithm Specification

- Compound data types can be formed with records.

```
Node = Record
{
   data type-1 data-1;
   …
   data type-n   data-n;
   Node * link;
}
```

Algorithm Specification

- Assignment of values to variables is done using the assignment statement.

  <Variable> := <expression>;

- There are two Boolean values **true** and **false**. Hese values are produced by

  Logical Operators **and, or, not**

  Relational Operators **<, <=, >, >=, =, !=**

Algorithm Specification

- Aggregates like arrays are represented by [and].
- **`A[1:n]`** represents an array with elements whose indices range from 1 to n.
- **`A[1:m,1:n]`** represents a two-dimensional array.
- **`A[i] and A[i,j]`** refer the elements.

Algorithm Specification

- A conditional statement has the following forms.

```
if (<condition>) then <statement-1>

if (<condition>) then <statement-1>
else <statement-2>
```

Algorithm Specification

- Case statement

```
case
{
   : <condition-1> : <statement-1>
   …
   : <condition-n> : <statement-n>
   : else: <statement-n+1>
}
```

Algorithm Specification

- Loop Statements:

```
while (< condition >) do
{
   <statement-1>
   …
   <statement-n>
}
```

Algorithm Specification

- Loop Statements:

```
for  <var>:=<val-1>  to  <val-2>  step
<step> do
{
   <statement-1>
   …
   <statement-n>
}
```

Algorithm Specification

- Loop Statements:

```
repeat
{
   <statement-1>
   …
   <statement-n>
}until (<condition>)
```

Algorithm Specification

- Input and output are done using the instructions **read & write**.
- There is only one type of procedure known as Algorithm.
- The Algorithm takes the form of a header followed by its block.

Algorithm Name (Parameter lists)

Algorithm Specification

```
Algorithm Name ( Parameter list )
{
    <stmt-1>
    …
    <stmt-2>
}
```