CS403: Algorithm Design and Analysis

Feb-Jun 2017

Indian Institute of Technology Mandi

Assignment 5

Priyansh Saxena

B14118

Computer Science and Engineering

CS 403 – Algorithm Design and Analysis

Assignment 5

Problem 1: Implement the algorithm to find k-node vertex cover of a given graph:

```
To search for a k-node vertex cover in G:
  If G contains no edges, then the empty set is a vertex cover
  If G contains> k |V| edges, then it has no k-node vertex cover
  Else let e = (u, v) be an edge of G
    Recursively check if either of G−{u} or G−{v}
                has a vertex cover of size k − 1
    If neither of them does, then G has no k-node vertex cover
    Else, one of them (say, G−{u}) has a (k − 1)-node vertex cover T
       In this case, T ∪ {u} is a k-node vertex cover of G
    Endif
  Endif
```

The running time of the Vertex Cover Algorithm on an n-node graph, with parameter k, is $O(2^k \cdot kn)$.

1. The time complexity can be found by solving a recurrence of the form T(n,k) where n is the number of vertices in the graph and k is the input parameter.
2. The base case of the recurrence is when the value of k is 1. In this case the time complexity is O(n).
3. The function calls itself twice with the value of k reduced by 1 in the general case and an additional time of O(n) is spent in each call. So the recurrence can be written as follows :
   $T(n, 1) \leq cn,$
   $T(n, k) \leq 2T(n - 1, k - 1) + ckn$
4. Therefore by induction on k, it can be shown that $T(n,k) \leq c.2^k kn$.

The time-complexity of the implementation is higher than this value because of the use of adjacency-list representation of graph and sets.

The space-complexity of the solution is O(n + m).

Problem 2: Implement the algorithm to find a maximum-size independent set in a forest.

```
To find a maximum-size independent set in a forest F:
  Let S be the independent set to be constructed (initially empty)
  While F has at least one edge
     Let e=(u,v) be an edge of F such that v is a leaf
     Add v to S
     Delete from F nodes u and v, and all edges incident to them
  Endwhile
  Return S
```

The time complexity of the implementation is $O(m^2n)$, which can be reduced to O(m) by storing the graph as an adjacency list using unordered maps, taking constant time for each of the m edges deleted.

The space-complexity of the implementation is O(m+n).

Problem 3: Implement algorithm to find a maximum-weight independent set of a tree.

To find a maximum-weight independent set of a tree T:

  Root the tree at a node r

  For all nodes u of T in post-order

    If u is a leaf then set the values:

$$M_{out}[u] = 0$$

$$M_{in}[u] = w_u$$

    Else set the values:

$$M_{out}[u] = \sum_{v \in children(u)} \max(M_{out}[u],\ M_{in}[u])$$

$$M_{in}[u] = w_u + \sum_{v \in children(u)} M_{out}[u].$$

    Endif

  Endfor

  Return $\max(M_{out}[r], M_{in}[r])$

The algorithm visits each vertex of the tree exactly once and the time spent to calculate the value of M_{in} and M_{out} for a particular vertex is amortized O(1), the time complexity of the overall algorithm is O(n), where n is the number of vertices in the graph.

The space-complexity of the implementation is O(m+n).