Aaryan Ajay Sharma 2022121001

04/11/2022

Algorithm Analysis & Design

Problem set 5

1. a) Let V_1 contain all vertices of even number of levels (level 0, level 2, ..) of the tree.

Let V_2 contain all the odd number of levels (level 1, level 3 ...)

There cannot be an edge from V_1 to itself as a vertex of level m can only can only be neighbours with ~~vertex~~ vertices of level m-1 & m+1

Similarly, the same argument can be made for vertex set V_2.

Therefore every edge has one vertex in V_1 and the other in V_2.

∴ Thus the tree is always a bipartite graph.

b) Suppose G_1 has an odd cycle.
Then obviously it cannot be bipartite, because no odd cycle is 2-colorable.

Conversely, suppose G_1 has no odd cycles.
Then we can only colour the vertices greedily by 2-colours, always choosing

a different color for a neighbours of same vertex which has been greedily colored already.

Any edge (additional) are consistent with our colouring, otherwise they would close a cycle of odd length with the edges we considered already.

The easiest external question is about the maximum possible number of edges in a bipartite graph on n vertices.

⇒ Bipartite graph: A graph which is 2-colorable is called bipartite.

⇒ Bipartite graphs, by definition, have chromatic number of 2 (Given it's not null).

c) Algorithm:

1. Run BFS of G & Build the BFS tree T.

2. Insert even level vertices ~~with~~ in empty set V_1 & odd level vertices in set V_2.

3. Check if there are any edges between ~~set~~ vertices of set V_1 & V_2 (i.e. from V_1 to V_1 & from V_2 to V_2)

2022/2/001 __/__/____

4. If not, the graph is bipartite. Else it is not.

2. Suppose that T is a minimum spanning tree of G and T' is spanning tree with a lighter bottleneck edge. Thus, T contains an edge e that is heiner them every edge in T'. So if we add e to T', it forms a cycle C on which it is the heiviest edge (since all other edges in C belong to T'). By the cut property, then e does not belong to any minimum spanning tree, contradicting. The fact that it is in T and T is a minimum spanning tree.

But the vice-versa is not true.

Let G have vertices $\{v_1, v_2, v_3, v_4\}$ with edges between each pair of vertices and with the weight on the edge from v_i to v_j equal to $i+j$. Then every tree has a bottleneck edge of weight at least 5.

So the tree consisting of a path through vertices v_3, v_2, v_1, v_4 is a minimum bottleneck tree. It is not a minimum spanning tree, however. Since its total weight is (remember that) that of the tree with edges from v_1 to every other vertex.

20221210 01

__/__/__

(Divide & Conquer)

3. Split the larger number into $\lceil n/m \rceil$ chunks, each with m digits. Multiply the smaller number by each chunk in $O(m^{\log 3})$ time using Karatsuba's algorithm, and then add the resulting partial products with appropriate shifts.

```
SKEWMULTIPLY(x[0...m-1], y[0....n-1]):
    prod ← 0
    offset ← 0
    for i ← 0 to ⌈n/m⌉ - 1
        chunk ← y[i·m ... (i+1)m-1]
        prod ← prod + MULTIPLY(x, chunk) · 10^(i·m)
    return prod
```

Each call to MULTIPLY requires $O(m^{\log 3})$ time, and all other work within a single iteration of the main loop requires $O(m)$ time.

Thus, the overall running time of the algorithm is $O(1) + \lceil n/m \rceil O(m^{\log 3}) = O(m^{\log 3 - 1} n)$ as required.

This is the standard method for multiplying a large integer by a single "digit" the integer written in base 10^m, but each single-"digit" multiplication implemented using Karatsuba's algorithm.

20221210 01

__/__/____

Dynamic Programming

4. Let LISS(x) indicates the size of largest independent set of a tree with root x.

LISS(X) = Max { (1 + sum of LISS for all grandchildren of X), (sum of LISS for all children of X) }.

→ The following recursion makes sense as a node which a part of LIS, then its children cannot be part of LIS, but its grandchildren can be.

We can build an array of size i-1, and from these we can calculate LISS(i).

∴ No. of unique sub-problems = O(n).

∴ Time complexity = O(n).

Network Flows

5. Algorithm a create an array A

1. Find a minimum (s-t) cut M. Using the procedure described at the end. Record len(M) in A.

2. Increase the capacity of any edge in M

by 1.

Now, if the min cut was unique in G, the max flow of G would have changed

3. Find the max flow of the new graph $= B'f'$

If $f' = f$ the min cut is unique

If $f' = f$ then min cut is not ~~max~~ unique
($\exists$ another min cut which gave $f' = f$)

4. If there is a new min-cut, then find it using 1.

Record $len(M_i)$ in A
$A.push(M_i)$

5. Keep doing this till the condition in 3 does not show that there is remaining min cut.

6. Loop through A to get $M_{i_{min}}$ (the minimum value value) in $O(n)$
Return $M_{min} = \min_j (j \in M \ \& \ M \text{ is min})$

Procedure to find:-

1. Run ford fulkerson algorithm in polynomial time $O(|f^*|E)$, and consider the final residual

2022121001

__/__/____

graph G.

2. Use BFS to find the set of vertices that are reachable from the sources in $Be = m \cdot O(m+n)$

Let $B = V - A$

Then (A, B) is a min cut of G.

Time $= O(m(|f^*|E| + m^2(m+n) + mn)$ ← upper ~~bound~~ for no. of min cuts.

$=$ Polynomial Time.

P v NP

1. Given a solution to $N^2 \times N^2$ sodoku we can check if it is valid or not by checking if the following three conditions are met:

1) Each row contains unique values from 1 to N^2.

2) Each unique values from 1 to N^2.

3) Each of the N^2 sized sub-squares of size $N \times N$, contains a unique value from 1 to N^2.

All of this can be done in $O(N^c)$ where $c \in N$.

$\therefore$ Sudoku $\in$ NP.