Hw1 pg1

Wednesday, April 6, 2022 11:12 AM

Exercise 1 (3 pts)
Let $G = (V, E)$ be any undirected graph. Recall that $\deg(v)$ gives the *degree* of $v \in V$ (which is the number of edges incident to v). Argue that

$$\sum_{v \in V} \deg(v) = 2 \cdot |E|.$$

consider two arbitrary vertices $u, v \in V(G)$ with edge $e_u \in V(E)$ where $e_u = \{u, v\}$ i.e. e_u is edge from u to v

Then $\deg(u) = \sum$ edges connected to u: $e_u = \{u, v\}$

Now consider the vertex v from above ($v \in \{u, v\}$). Clearly $\exists\, e_u = \{u, v\} \Rightarrow \exists\, e_v = \{v, u\} = e_u$ by symmetry.

Thus, an edge $e_u = e_v$ is counted twice when calculating $\sum_{u \in V(G)} \deg(u) \Rightarrow \sum_{u \in V(G)} \deg(u) = 2 \cdot |E| = 2 \cdot |E(G)|$ $\square$ (hopefully)

Exercise 2 (8 pts)
Let $T = (V, E)$ be a graph that is a tree and that has $|V| = n$ nodes and assume that $n \geq 2$.

i) Show that T has at least 2 vertices of degree 1 (also called *leaves*).

ii) Use i) to prove by induction that the tree has exactly $|E| = n - 1$ many edges.
Remark: This quantity also falls out of another proof that we will see in the lecture. But please give your own proof by induction here.

iii) Show that T has at most $\frac{n}{2}$ many vertices that have degree 3 or higher.

i) Consider two vertices v_{i-1}, v_i where $i = \{2, \dots, n\}$ by Assumption $n \geq 2$

For the case $n = 2$, $v_{i-1}, v_i = v_1, v_2$ have edge $e = \{v_1, v_2\}$ by definition of tree is (connected)

Therefore edge $e \Rightarrow \deg(v_1) = 1$ and $\deg(v_2) = 1 \Rightarrow T$ has at least 2 vertices with degree 1

ii) Prove by induction that Tree T has exactly $|E| = |E(T)| = n - 1$ edges

Base case: $n = 2$

By part i) $n = 2 \Rightarrow \exists$ edge e connecting the 2 vertices (these are the leaves with degree

$n = 2 \wedge T$ acyclic by definition $\Rightarrow e$ is the only edge in $T \Rightarrow |E|$ $|E(T)| = 1 = 2 - 1 = n - 1$

Induction case: $n \geq 3$

Consider default Tree T_n with n nodes. $V(T_n) = \{v_1, v_2, \dots, v_n\}$

By part i) $\exists\, v_n \in T_n$ s.t. $\deg(v) = 1$; v_n is a leaf.

Let us keep track of $|E(T_n)|$ with variable total_sum = 0 = for each vertex

e.g. degree$(v) = 1 \Rightarrow$ total_sum = total_sum + 1 (adding $\deg(\text{leaf}) = 1$ to total sum)

Consider the following algorithm to find $|E|$:

```
total_sum = 0    } initial set up
T_prime   = T_n  }

while |V(T_prime)| ≥ 2

        total_sum = total_sum + 1   (adding deg(leaf) = 1 to total_sum)
        Remove current leaf from T_prime
```

while $|V(T\_prime)| \geq 2$ i.e. while there still exists an edge in T_prime, by $T\_prime = T_n$ initially $\wedge\, T_n$ is tree (connected); stop when $|V(T\_prime)| = 1$ b/c 1 vertex $\Rightarrow$ 0 edges remaining

Remove current leaf from T_prime $\leftarrow$ Key step: trimming leaf from T_prime decrements the number of edges that still need to be counted by 1 because $\deg(\text{leaf}) = 1$

This is equivalent to summing the number of leaves in T_n while T_prime is not the singular vertex = v_{root}

therefore $|E(T_n)| = \left|\{\text{leaves}(T_n)\} \setminus v_{root}\right| = |V(T_n)| - |v_{root}| = n - 1$ $\square$ (hopefully)

iii) Show $\left|A = \{a \in V(T) \mid \deg(a) \geq 3\}\right| \leq \frac{n}{2}$

By exercise 1, $\sum_{u \in E(T)} \deg(u) = 2|E(G)|$ for any undirected graph $G \Rightarrow$ for T $\underbrace{\sum_{u \in V(T)} \deg(u)}_{= 2|E(T)|} = 2(n-1)$ by part ii)

We can then write $2|E(T)| = \underbrace{\left(\sum_{a \in A} \deg(a) + \sum_{v \in V(T) \setminus A} \deg(v)\right)}_{\text{split } T \text{ into 2 groups to get desired value } |A|} \underset{\text{b/c } \forall a \in A\ \deg(a) \geq 3}{\geq} \left(\underbrace{3|A|} + (|V(T)| - |A|)\right) = 2|A| + |V(T)| = 2|A| + n$

Then $2|E(T)| = 2(n-1) \geq 2|A| + n \Rightarrow |A| \leq \frac{2n - 2 - n}{2} = \frac{n-2}{2} \leq \frac{n}{2} \Rightarrow |A| \leq \frac{n}{2}$ $\square$ (but I'm a bit confused on when $|A| = \frac{n}{2}$)

Hw1 pg2

Wednesday, April 6, 2022 2:16 PM

Exercise 3 (9 pts)
In the lecture we saw that given a complete graph $K_n = (V, E)$ with edge cost $c_{ij} \geq 0$ for $\{i, j\} \in E$ one can compute a minimum cost TSP tour in time $O(2^n n^3)$ using dynamic programming. Here, we want to consider a variant of this problem:

> INPUT: Complete graph $K_n = (V, E)$ on n vertices with edge cost $c_{ij} \geq 0$ (for $\{i, j\} \in E$) and a parameter $m \in \{1, \ldots, n\}$.
> GOAL: Find a minimum cost cycle in K_n that connects exactly m nodes.

Give an algorithm (based on the dynamic program for TSP) that solves the problem. Which running time do you get? (a straightforward solution would get the minimum of $O(n^3 2^n)$ and $O(n^3 n^m)$ — the latter bound is better if m is a lot smaller than n). Assume $n, m \geq 3$ .. Recall Held-Karp Algorithm

$C(\{i,j\}, i, j) := c_{i,j}$ for all $i \neq j$

For $k = [3, m]$: Stop at $K = n$ because we want output length n

    For all sets $S \subseteq V$ such that $|S| = K$ AND $i, j \in S$:

        $C(S, i, j) := \min_{l \in S \setminus \{i,j\}} \{ C(S \setminus \{j\}, i, l) + c_{l,j} \}$

    End

End

$$\text{output: } \min \Big\{ \underbrace{C(A, i, j)}_{\text{cost from start to end}} + \underbrace{c_{j,i}}_{\text{cost of returning to start node (complete the cycle)}} \;\Big|\; A \subseteq V, \underbrace{|A| = m}_{\text{goal of connecting exactly } m \text{ nodes}}, i, j \in A \Big\} \quad \text{(such that)}$$

My thinking (unsure if correct):
(min cost path from i to l using only nodes in $S \setminus \{j\}$ + cost of edge $\{l, j\}$)
$\approx$ i.e. best path from i to j is minimum of previously calculated path from i to another vertex l + $c_{l,j}$ ... find best l

Running Time:

Initialize costs of all edges: $O(n)$

Outer For Loop Runtime:

$K = [3, m]$: $O(m)$; $m \leq n \Rightarrow O(n)$

Inner For loop Runtime:

for all subsets $S : |S| = K \Rightarrow n \cdot \binom{n}{K}$ subsets $\approx O(n \cdot n^m)$ → operate over n elements to find subsets m times

do stuff on $\approx |S| = K$ elements.

$\Rightarrow O(m)$; $m \leq n \Rightarrow O(n)$

$\Rightarrow O(n \cdot n^m) \cdot O(n)$

Nested for loop $\Rightarrow O(n) \cdot O(n^2 n^m) \Rightarrow O(n^3 n^m)$

Calculate output min: n^m possible subsets in A; n^2 to iterate over $i, j \in A$

calculate $C(A, i, j) + c_{i,j}$ n times

$\approx O(n^3 n^m)$

$\Rightarrow$ Total Runtime has complexity $O(n^3 n^m)$