Math 182 HW1 Jun Ryu

Section 2

1) a) a) $(2n)^2 = 4n^2$: slower by a factor of 4

b) $(n+1)^2 = n^2+2n+1$: slower by an additive $(2n+1)$

b) a) $(2n)^3 = 8n^3$: slower by a factor of 8

b) $(n+1)^3 = n^3+3n^2+3n+1$: slower by an additive $(3n^2+3n+1)$

c) a) $(100(2n)^2) = 400n^2$: slower by a factor of 4

b) $(100(n+1)^2) = 100n^2+200n+100$: slower by an additive $(200n+100)$

d) a) $2n\log 2n$: slower by a factor of 2 and an additive $2n$.

b) $(n+1)\log(n+1)$: slower by $(n+1)\log(n+1) - n\log n$

e) a) 2^{2n} : squares the original running time

b) 2^{n+1} : slower by a factor of 2

4) In order of increasing order of growth rate:

$g_1, g_5, g_3, g_4, g_2, g_7, g_6$

Brief justifications:

we have $g_1 = O(g_5)$ since we are comparing $\sqrt{\log n}$ to $\log n$.

we have $g_5 = O(g_3)$ since $(\log n)^3$ grows much faster than $\log n$

we have $g_3 = O(g_4)$ since logarithms will grow slower than polynomials

we have $g_4 = O(g_2)$ since polynomials grow slower than exponentials

we have $g_2 = O(g_7)$ since we are comparing the exponents to be n versus n^2.

we have $g_7 = O(g_6)$ since 2^n will grow a lot faster than n^2.

5) a) False: counterexample:

let $g(n) = 1$, then we have $\log_2 g(n) = 0$.

If $\log_2 f(n) > 0$, then the following inequality: $\log_2 f(n) \leq c \cdot \log_2 g(n)$, $\left(\begin{matrix}\forall n, \text{ for} \\ \text{some } c > 0\end{matrix}\right)$

can never be satisfied. So, let $f(n) = 2$

$\Rightarrow \log_2 f(n) = 1 > 0$.

b) False: counter example:

If $f(n) = 2n$ and $g(n) = n$, then

we have $4^n \stackrel{?}{=} \Theta(2^n)$, which

is not satisfied as

$4^n \not\leq c \cdot 2^n$ for some $c > 0$

c) True: Since we were given that

$f(n) \leq c \cdot g(n)$ for some $c > 0$,

then we have $(f(n))^2 \leq c^2 \cdot (g(n))^2$ for

the same c, $\forall n$.

7) Suppose it takes L lines to fulfill n words. We write the pseudocode as following:

```
Initialize lines 1, 2 ... L
For i = 1 ... L;
    For j = 1 ... i;
        Sing line j, line j-1 ... line 1
    end
end
```

Here, since n is the number of words:

we have $1+2+\cdots+L \le n$

$$\Rightarrow \frac{(L-1)^2}{2} \le \frac{L(L-1)}{2} \le n$$

$$\Rightarrow L \le 1+\sqrt{2n} = O(\sqrt{n}) = f(n)$$

8) a) We use the first jar to test from heights as the following: $\lfloor\sqrt{n}\rfloor, \lfloor 2\sqrt{n}\rfloor, \lfloor 3\sqrt{n}\rfloor \ldots$ until it breaks. If the 1^{st} jar breaks at some point in the middle, (suppose $\lfloor s\sqrt{n}\rfloor$)

we know the highest safe rung is in between $\lfloor s\sqrt{n} \rfloor$ and $\lfloor (s-1)\sqrt{n} \rfloor$.

So, now we start from $\lfloor (s-1)\sqrt{n} \rfloor$ and go up by 1 each time.

(i.e. test $\lfloor (s-1)\sqrt{n} \rfloor + 1$, $\lfloor (s-1)\sqrt{n} \rfloor + 2 \ldots$)

With this method, the first jar will take at most $\sqrt{n}$ drops and the same applies for the second jar. So, in total, $2\sqrt{n}$ drops are performed $\Rightarrow O(\sqrt{n})$

$\Rightarrow$ slower than linear time

b) We slightly modify part a). We drop the first jar from heights $\lfloor n^{\frac{k-1}{k}} \rfloor, \lfloor 2n^{\frac{k-1}{k}} \rfloor \ldots$

and then now we drop the second jar within that one interval of $n^{\frac{k-1}{k}}$ where the first jar broke. Note that this will only require $2(k-1)n^{1/k}$ drops in total and the first jar will take a maximum of $2n^{1/k}$ drops, giving us an upper bound of $2kn^{1/k}$ for $f(n)$.

We notice that if $f_k(n) \le 2kn^{1/k}$, as k increases, f_k will grow asymptotically slower than all its previous functions since k is part of the exponent.

Section 3

2) We perform a BFS starting from any arbitrary vertex v, which results in a tree. Now, we try to locate the edges in the original graph that are <u>not</u> included in our resulting tree. From this, we can locate our cycle in $O(m+n)$ time by connecting the edge from the original graph with ≥ 2 adjacent edges found in the tree.

5) Base case: A Tree with 1 node:

# of nodes with 2 children : 0

# of leaves : 1

So the claim holds ✓

Induction step: Now, assume T is a binary tree with # of nodes > 1 and let v be a leaf. v has a parent, which we'll denote by u. We observe what happens when v is deleted and the resulting tree is called T^*. If u had no other leaf attached, it now becomes a leaf in T^* so the # of leaves stay the same and so does the # of nodes with 2 children. If u had another leaf, # of leaves will be subtracted by 1 from T, but so will the # of nodes with 2 children, thus not changing the result. In both cases, by applying the inductive hypothesis to T^*, we conclude the induction.

6) Proof by contradiction:

Suppose $\exists$ an edge $e = \{a, b\}$ in G that is not part of the tree T.

As T is a DFS tree, one point must be an ancestor of the other. As T is a BFS tree as well, the distance of a and b from another point c in T can only differ by at most 1. However, if a is an ancestor of b WLOG, and $dist(b, c)$ is at most 1 greater than $dist(a, c)$, it means a is a direct parent of b $\Rightarrow$ contradiction since then, e would be part of the tree

7) This is true. Let G be as described in the problem. We will prove by contradiction. Assume G is not connected. We let S be the nodes in the smallest connected

part. Since there are at least 2 connected parts, $|S| \le n/2$. Now, pick any arbitrary node $v \in S$. Its degree can at most be $n/2 - 1$, which is less than $n/2$
$\Rightarrow$ contradicts the assumption that every node has degree at least $n/2$.

8) This is false. Consider the following graph: We have nodes $u_1, \dots u_{k-1}$ and a path that crosses through them. We also have another set of nodes $v_1 \dots v_{n-k+1}$ where each one is separately connected to u_1 by a single edge. Now, we try to put an upper bound to apd(G). With at least 1 point from $u_1 \dots u_{k-1}$, there are kn 2-element combinations that result in a maximal distance of k. The other cases will

only have a maximal distance of 2.

Thus, $\text{apd}(G) \le \dfrac{2\binom{n}{2} + (kn)n}{\binom{n}{2}} = 2 + \dfrac{2k^2}{n-1}$

Now, if we choose n s.t. $n-1 \ge 2k^2$, we have $\text{apd}(G) < 3$ and if $k > 3c$, then we have $\text{diam}(G) > 3c$ so

$\dfrac{\text{diam}(G)}{\text{apd}(G)} > \dfrac{3c}{3} = c$ so false.

9) Suppose we perform BFS from node s. Since $\text{dist}(s,t) > n/2$, we claim that one of the layers in between $L_1 \ldots L_{d-1}$ only has a single node. We know this is true because if all of them have more than 1, this would have at least $2(n/2) = n$ nodes but G has n nodes in total. Thus, for that layer with a singular node, call the node v.

If v is cut from G, there is no way to go from S to T since the BFS tree would be disconnected.

Thus, an algorithm that would suffice would be to consider a set of nodes $\{s\} \cup L_1 \cup \cdots \cup L_i$ and to check for a layer with a single node. That node, found in L_i through iteration will be v.