Problem Set 1

All parts are due February 18th, 2016 at 11:59PM.

Name: Alex List

Collaborators: /

Part A

Problem 1-1.

(a) $f2 = log(4n^{n^4}) = n^4 log(n)$
$f2 > f1 \rightarrow f1 = O(f2)$
$f3 = 4log(4n)log(n) \simeq log(n)log(n) = log^2(n)$
$\lim n \rightarrow \inf f3/f2 = \lim n \rightarrow \inf log^2(n)/n^4 log(n) = \lim n \rightarrow \inf log(n)/n^4 = 0$
$\rightarrow f3 = O(f2)$
$\lim n \rightarrow \inf f3/f4 = \lim n \rightarrow \inf log^2(n)/log^4(n) = 0 \rightarrow f3 = O(f4)$
$\lim n \rightarrow \inf f5/f4 = \lim n \rightarrow \inf log^4(log(n))/log^4(n) = 0 \rightarrow f5 = O(f4)$
$\rightarrow f1 < f2 > f3 < f4 > f5$
Comparing $f1\&f3$
$\lim n \rightarrow \inf f3/f1 = \lim n \rightarrow \inf log^2(n)/n^4 = 0 \rightarrow f3 = O(f1)$
$\rightarrow f3 < f1 < f2\&f5 < f4 > f3$
Comparing $f2\&f5$
$\lim n \rightarrow \inf f5/f2 = \lim n \rightarrow \inf log^4(log(n))/n^4 log(n) = \lim n \rightarrow \inf log(log(n))/nlog(n) = 0 \rightarrow f5 = O(f2)$
$\rightarrow f3 < f1 < f2\&f2 > f5 < f4 > f3$
Comparing $f3\&f5$
$\lim n \rightarrow \inf f5/f3 = \lim n \rightarrow \inf log^4(log(n))/log^2(n) = \lim n \rightarrow \inf log(log(n))/log(n) = 0 \rightarrow f5 = O(f3)$
$\rightarrow f5 < f3 < f1 < f2\&f4 > f3\&f4 > f5$
Comparing $f4\&f1$
$\lim n \rightarrow \inf f4/f1 = \lim n \rightarrow \inf log^4(n)/n^4 = 0 \rightarrow f4 = O(f1)$
$\rightarrow f5 < f3 < f4 < f1 < f2$
$f5, f3, f4, f1, f2$

(b) Transform each function with logarithm and compare:
$f'1 = log(4^{4^n}) = 4^n log(4)$
$f'2 = log(4^{4^{n+1}}) = 4^{n+1} log(4)$
$f'3 = log(5^{4^n}) = 4^n log(5)$
$f'4 = log(5^{4n}) = 4n log(5)$
$f'5 = log(5^{5n}) = 5n log(5)$
By inspection:
$f4 = O(f5), f1 = O(f2), f4 = O(f3), f1 = O(f3), f3 = O(f2), f5 = O(f3), f5 = O(f1), f4 = O(f1)$
$f4 < f5, f1 < f2, f1 < f3, f3 < f2, f4 < f3, f5 < f3, f5 < f1, f4 < f1$
$\rightarrow f4 < f5 < f1 < f3 < f2$
$f4, f5, f1, f3, f2$

(c) Simply first then compare
$f1 = \frac{n!}{4!(n-4)!} = O(n^4)$
$f2$ via Sterling's approximation can be ln'd $ln(f2) = \ln(n!) = n\ln(n) - n + O(\ln(n)) \simeq O(nln(n))$ Compare $ln(f2)$ as $O(nlnn)$
$f2 = \frac{n!}{(n/4)!(n-n/4)!} = \frac{n!}{(n/4)!(3n/4)!} = \frac{(n)(n-1)(n-2)*...*(3n/4)}{(n/4)(n/4-1)(n/4-2)*...*(1)} = O(n^{n/4})$ Yet probably faster
$f3 = 4n! = O(n^n)$
$f4 = 4^{n/4}$
$f5 = (n/4)^{n/4}$
Visually:
$f2 < f1 < f3, f5 = O(f3), f1 = O(f5), f2 = O(f4)$
$\rightarrow f2 < f1 < f5 < f3, f2 = O(f4)$
I'll compare f4 and f3. I wasn't sure, so I use L'Hospital's rule
$\lim n \rightarrow \inf \frac{d}{dn} \frac{f4}{f3} = \lim n \rightarrow \inf \frac{d}{dn} \frac{4^{n/4}}{n^n} = \lim n \rightarrow \inf \frac{2^{n/2-1} log(2)}{n^n(log(n)+1)} = 0$
$\rightarrow f4 = O(f3)$
I still need to compare f4 and f5. I'll use L'Hospital's rule again.
$\lim n \rightarrow \inf \frac{d}{dn} \frac{f5}{f4} = \lim n \rightarrow \inf \frac{d}{dn} \frac{(n/4)^{n/4}}{4^{n/4}} = \lim n \rightarrow \inf \frac{2^{-n/2-2} n^{n/4} log(n/4+1)}{2^{n/2-1} log(2)} = 0$
$\rightarrow f5 = O(f4)$
Because the $2^{-n/2-2}$ term goes to 0.
$\rightarrow f2 < f1 < f5 < f4 < f3$
$f2, f1, f5, f4, f3$

Problem 1-2.

(a) 1. $\theta(n)$ Branch factor 1. N iterations of c work.

2. $O(n^2)$ Because summation over work on all levels from c to nc is $\frac{n*(n+1)}{2} = \frac{n^2+n}{2} = O(n^2)$
3. $\theta(log(n))$ Because $log(n)$ levels with c work.
4. $O(n)$ Because with branching factor 2, work c per node, bottom row does most work, $c * 2^{height=log(n)} = cn = O(n)$
5. $\theta(nlog(n))$ Because there are $log(n)$ rows of cn work per row.
6. $O(n^{log(3)})$ Because with branching factor 3, the bottom row does most work $3^{height=log(n)}$ nodes of c work $= c * n^{log(3)/log(2)} = O(n^{log(3)})$

(b) 1. $T(n) = T(n/2) + c$ Binary search has one subproblem, with half the nodes to visit. $O(log(n))$ tota.

2. $T(n) = T(n/2) + c * log(n)$ For nxn matrix, Instead of doing c work for 1 comparison, I search the n-width row, $log(n)$ work per subproblem.

Problem 1-3.

(a) The nieve solution is $O(n^4)$ with 4-nested loops. It holds a variable named $maxGain$ while (for abuy1 in A for first buy date (for asell1 $>$ abuy1 in A for first sell date (for abuy2 $>=$ asell1 in A for second buy date (for asell2 $>$ abuy2 in A for second sell date (maxGain = A[asell1] - A[abuy1] + A[asell2] - A[abuy2] if A[asell1] - A[abuy1] + A[asell2] - A[abuy2] $>$ maxGain))))

```
ans = 0
for b0 in range(n):
  for s0 in range(b0,n):
    for b1 in range(s0,n):
      for s1 in range(b1,n):
        ans = max(ans, A[asell1] - A[abuy1] + A[asell2] - A[abuy2])
return ans
```

(b)

Part B

Problem 1-4.

Submit your implemented python script.

(a)

(b)

(c)

(d) Part A: $O(n)$

1. Create a new dictionary to maps words to array containing the word's count for each word list $O(m)$
   (a) Iterate through each word list $O(1)$
   (b) Iterate through each word in list $O(n)$
   (c) Add frequency of word in current iterating list's $wordList$ to the $word$'s dictionary entry, at the column for the current iterating list $O(1)$
2. Get dot-product over dictionary's frequency values for each word $O(m)$

Part B: $O(n)$

1. Create a new dictionary to maps words to array containing the word's count for each word list $O(m)$
   (a) Iterate through each word list $O(1)$
   (b) Iterate through each word in list $O(n)$
   (c) Add frequency of $word + nextWord$ in current iterating list's $wordList$ to the $word + nextWord$'s dictionary entry, at the column for the current iterating list $O(1)$
2. Get dot-product over dictionary's frequency values for each word pair $O(m + n)$ m if all pairs the same, up to n if all pairs different

Part c: $O(n + mlogm)$ $mlogn$ may be greater than n, for example if no words occur twice.

1. get the frequency of the words for each list $O(n)$
2. Sort lists of words by frequency $O(mlogm)$
3. Truncate sorted word lists to 50 words per list $O(1)$
4. Create a new dictionary to maps words to array containing the word's count for each word list $O(k)$
   (a) Iterate through each truncated word list $O(1)$
   (b) Iterate through each word in list $O(k)$
   (c) Add frequency of word in current iterating list's $wordList$ to the word's dictionary entry, at the column for the current iterating list $O(1)$
5. Get dot-product over dictionary's frequency values for each word $O(k)$

(e) henry_iv_1 with

```
tempest doc_dist: 0.3929, pairs: 1.1059, dist_50: 0.3681
pirates doc_dist: 0.5333, pairs: 1.2576, dist_50: 0.5073
henry_iv_2 doc_dist: 0.3024, pairs: 0.9143, dist_50: 0.2901
```

Conclusions: doc_dist is the base case of accuracy. doc_dist_50 is almost as accurate, for a potential constant factor improvement in runtime. Pair distance is not a comparison method consistent with doc_dist.