Final

FINAL (2022-05-13)

See https://bit.ly/3KxuMdV for the official information on the time and location of this exam.

Question 1:

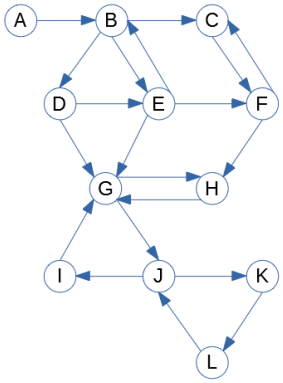
(A) Order functions in *increasing* order by their asymptotic growth rate: Write them in some sequence $g_1, g_2, \ldots, g_8$ such that $g_1 = O(g_2)$, $g_2 = O(g_3)$, ..., $g_7 = O(g_8)$.

$$f_1(n) = n^e, \quad f_2(n) = e^n, \quad f_3(n) = \binom{n}{n-3}, \quad f_4(n) = \sqrt{2^{\sqrt{n}}},$$
$$f_5(n) = \binom{n}{4}, \quad f_6 = 2^{\log_2^4 n}, \quad f_7(n) = n^{3(\log_2 n)^2}, \quad f_8(n) = (n-1)(n+1)(n^2+1)(n^4+1).$$

(B) Find function $g(n)$ such that $T(n)$ is in $\Theta(g(n))$. Here $T(n)$ is defined by a recurrence:

$$T(n) = \begin{cases} T\left(\lfloor \frac{n}{3} \rfloor\right) + T\left(\lfloor \frac{2n}{3} \rfloor\right) + n, & \text{if } n > 0, \\ 1, & \text{if } n = 0. \end{cases}$$

Question 2: Run Kosaraju algorithm to find strongly connected components.



(A) Run the DFS on the given graph. (Whenever you run out of traversable nodes and need to pick a new vertex, select the alphabetically first one. Also, when you visit the children of a given node – visit them in an alphabetical order.)

Label each vertex with two numbers `d/f`, where `d` is the discovery time, but `f` is the finishing time during the DFS traversal. (Both `d` and `f` are integers from the interval $[1; 2 \cdot 12] = [1; 24]$.)

(B) As requested by the Kosaraju algorithm, transpose the directed graphs matrix (i.e. flip all the arrows), and run the DFS again. (This time the ordering of the nodes is different: In Kosaraju algorithm they are determined by the `d/f` numbers found in the previous step. Use this ordering every time when you run out of traversable nodes or need to visit the children nodes.)

In this new graph mark all the strongly connected components you have found. (Components are marked by drawing an oval shape around all the nodes in that component.)

(C) Estimate the time complexity of this algorithm – find the smallest (slowest growing) $g(n)$ such that the runtime of the algorithm $T(n)$ is in $O(g(n))$.

Question 3: Consider the following sequence of hexadecimal digits:

$$T = \texttt{255044462D312E35}\ldots$$

The hash function is determined by the following formula:

$$h\left(a_0 a_1 a_2 \ldots a_{n-1}\right) = \left(\sum_{k=0}^{n-1} a_k 16^{n-1-k}\right) \bmod 109$$

(A) Select the window size $s = 6$ and find the hash values of the first two substrings ($P_0 = \texttt{255044}$ and $P_2 = \texttt{550444}$) in the given text.

(B) Describe the algorithm (using constant time) to skip a digit from the start of the pattern. Your algorithm can use the previous hash value and the window size s as inputs.

Run this algorithm to get $h(P_1) = h(\texttt{55044})$ from $h(P_0) = h(\texttt{255044})$ skipping the hex digit "2" at the very beginning (assuming that the window size is $s = 6$).

(C) Describe the algorithm (using constant time) to append a digit to the end of the pattern. Your algorithm can use the previous hash value and the window size s as inputs.

Run your algorithm to get $h(P_2) = h(\texttt{550444})$ from $h(P_1) = h(\texttt{55044})$. (Your hash value for $h(P_2)$ should coincide with the value obtained in (A).)

Question 4:

(A) Build the KMP (Knuth-Morris-Pratt) prefix function to search for this pattern: $P = \texttt{ababacab}$.

(B) Show how KMP works to find *all occurrences* of the pattern in the following text: $T = \texttt{abaababacababa}$.

Namely, write all the letters of this text in a horizontal line. Under this text show multiple copies of the pattern (copied with different offsets so that letters in the pattern are compared to the letters in the text T located directly above it). For each copy of the pattern circle those letters that were compared with the above text.

(C) How many letter-to-letter comparisons would be made, if P is searched in T using the naive search algorithm? How many were used by the KMP algorithm in (B)?

Question 5: Consider this list of strings:

```
S = ["Croatia", "Iceland", "Ireland", "Denmark", "Bulgaria", "Andorra"]
```

Insert this list into a hash table using the following hash function in Python:

```
hash('ABC') % 8
```

The above expression calculates the hash value for the string `'ABC'`. To make this hash function predictable, before you run the Python command-line or the IDE, set the environment variable using one of these commands:

```
export PYTHONHASHSEED=0
  OR
$Env:PYTHONHASHSEED=0
  OR
set PYTHONHASHSEED=0
```

(A) Draw the contents of hash table (eight slots, 0 to 7), if the collisions are handled by linear probing.

(B) Assume we want to insert the seventh entry. This entry is yet unknown and the Python's hash function can take every remainder modulo 8 with the same probability. What is the expected number of comparisons before this entry can be inserted into the hash table? (If the entry inserts into an empty slot, it is one comparison; if it needs to do one linear probing to move to the next cell,it is two comparisons; if it does two linear probings, then it is three comparisons, etc.)

(C) Before the seventh entry was inserted, the hash table was considered full and its size was doubled from 8 to 16 slots. (The new hash function is `hash('ABC') % 16`). Draw the new contents of the hash table containing the same six entries as before.