COS 226	Algorithms and Data Structures	Fall 2012

Final Exam Solutions

1. **Analysis of algorithms.**

   (a) 400 seconds

   (b) $\sim 4MR$

2. **Graphs.**

   (a) The method `marked[v]` returns true if and only if there is a directed path from s to v.

   (b) $E + V$, as usual for depth-first search.

   (c) V (to initialize the `marked[]` array).

   (d) V^2. Note that $E \leq V^2$ since there are no parallel edges.

3. **Graph search.**

   (a) reverse postorder: 0 1 6 5 2 8 9 4 3 7

   (b) preorder: 0 1 6 2 7 8 3 9 4 5

4. **Minimum spanning trees.**

   (a) 10 20 30 40 50 100

   (b) $x \leq 110$.

   (c) $y \leq 60$.

   (d) $z \leq 80$.

5. **Shortest paths.**

   (a) 0 1 5 4

   (b) $x = 8.0$

   (c) $y > 12.0$. (We also accepted $y \geq 12.0$.)

(d) vertex 2

(e)

v	distTo[]	edgeTo[]
3	20.0	$2 \to 3$
6	35.0	$2 \to 6$

6. **Maximum flow.**

(a) 25

(b) $A \to G \to B \to C \to H \to I \to J$

(c) $25 + 3 = 28$

(d) $\{A, B, C, F, G\}$

(e) 28

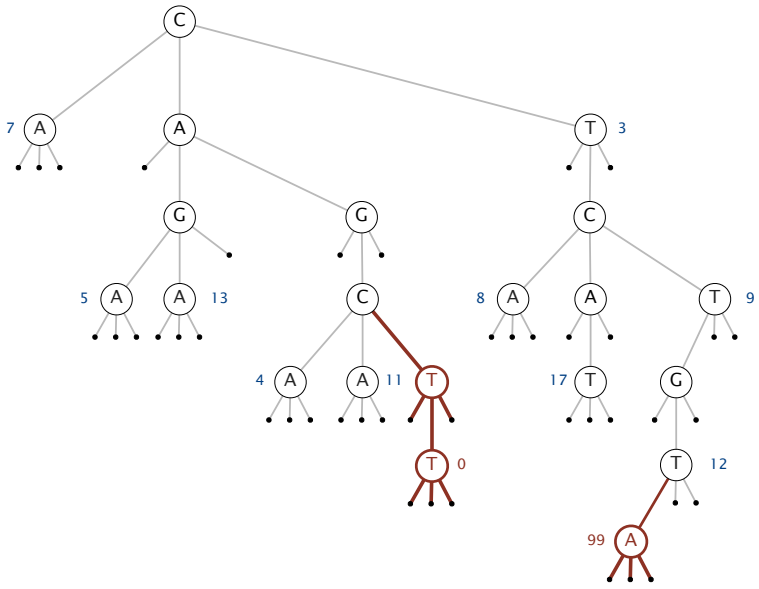
7. **String sorting algorithms.**

0 3 4 4 2 3 2 2 1

8. **Ternary search tries.**

(a) `A (7), CAA (5), CGA (4), CGCA (11), TA (8), TGT (12), TT (9)`

(b)

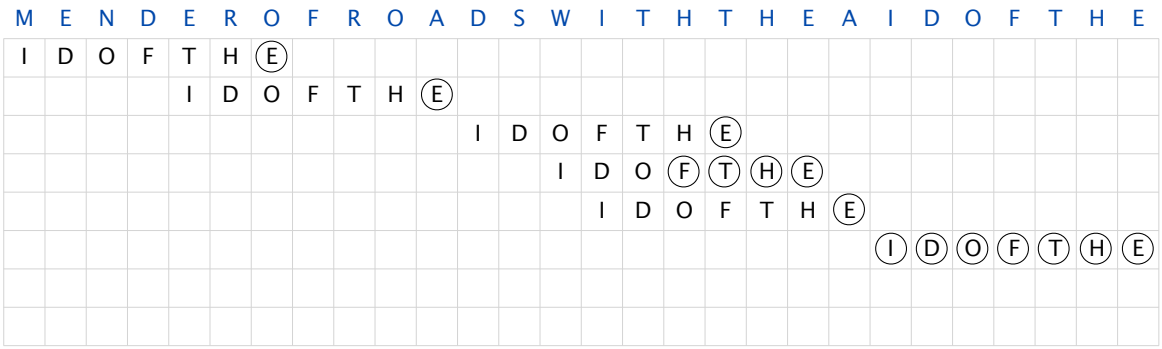


9. **Knuth-Morris-Pratt substring search.**

ABCABABABCAA

	0	1	2	3	4	5	6	7	8	9	10	11
A	1	1	1	4	1	6	1	8	1	1	11	12
B	0	2	0	0	5	0	7	0	9	0	0	5
C	0	0	3	0	0	3	0	3	0	10	0	0
s	A	B	C	A	B	A	B	A	B	C	A	A

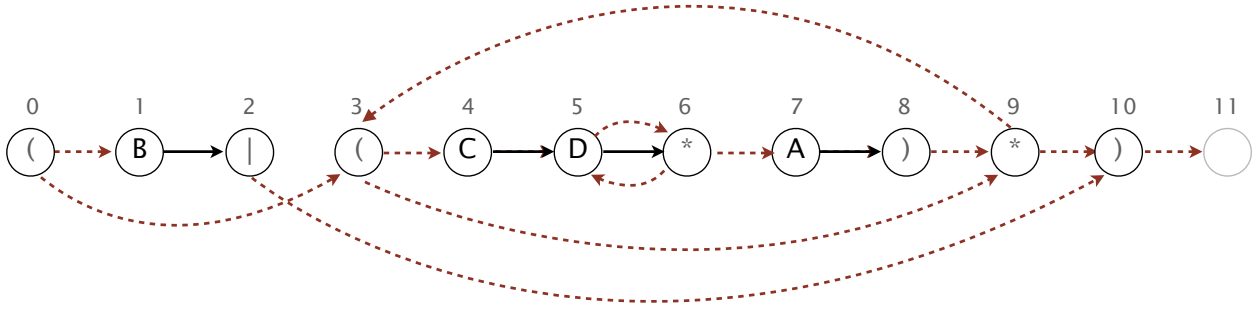
10. **Boyer-Moore substring search.**



11. **Regular expressions.**

$0 \to 3$ $2 \to 10$ $3 \to 4$ $3 \to 9$ $5 \to 6$ $6 \to 5$ $6 \to 7$ $9 \to 3$

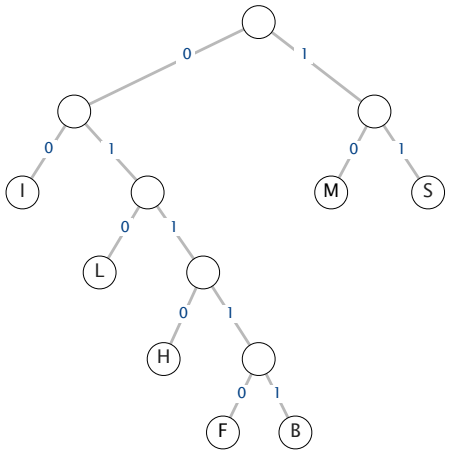
The edges $8 \to 9$, $9 \to 10$, and $10 \to 11$ are the remaining ϵ-transitions edges. Here is a drawing of the NFA.



12. **Huffman codes.**

(a)

char	freq	encoding
B	2	01111
F	1	01110
H	3	0110
I	?	00
L	5	010
M	15	10
S	15	11



(b) $6 \leq freq(I) \leq 15$.

- Since I is not touched until after merging L with $\{B, F, H\}$, $freq(I) \geq freq(L) = 5$ and $freq(I) \geq freq(\{B, F, H\}) = 2 + 1 + 3 = 6$.
- Since I is merged with $\{B, F, H, L\}$ instead of M or S, $freq(I) \leq freq(M) = 15$ and $freq(I) \leq freq(S) = 15$.

13. **Data compression.**

K X M R D S

$\frac{8}{255}$ Run-length coding with 8-bit counts for best case inputs of N bits.

The best case is an alternating sequence of 255 *0s and* 255 *1s. Each sequence of* 255 *0s (or* 255 *1s) is encoded with* 8 *bits* (11111111).

$\frac{8}{1}$ Run-length coding with 8-bit counts for worst-case inputs of N bits.

The worst case is an alternating sequence of 0s and 1s. Each bit is encoded with 8 bits (00000001).

$\frac{1}{8}$ Huffman coding for best-case inputs of N characters.

The best case is when one character occurs 100% of the time (or all but a constant number of times), in which case it is encoded using 1 *bit.*

$\frac{8}{8}$ *The worst case is when each of the* 256 *characters occurs with equal frequency. In this case, each character is encoded using* 8 *bits.*

$\frac{12}{8\times3840}$ LZW coding for best-case inputs of N characters using 12-bit codewords. Recall: no new codewords are added to the table if the table already has $2^{12} = 4096$ entries.

The best case is one 8*-bit character, say* `A`*, repeated* N *times. The table contains* 12*-bit codewords for* `A`*,* `AA`*,* `AAA`*, and so on, all the way up to* $2^{12} - 256 = 3840$ `A`*s when the table gets full. After this point, each sequence of* 3840 *consecutive* `A`*s is encoded using only* 12 *bits.*

$\frac{12}{8}$ LZW coding for worst-case inputs of N characters using with 12-bit codewords. Recall: no new codewords are added to the table if the table already has $2^{12} = 4096$ entries.

The worst case is when the codeword table gets filled up with useless codewords and then the rest of the message cannot take advantage of any of the added codewords. An input of 3840 *As followed by* $N - 3840$ *Bs would have this property. In this case, each B requires a* 12*-bit codeword.*

A. $\sim 1/4096$
B. $\sim 1/3840$
C. $\sim 1/2731$
D. $\sim 1/2560$
E. $\sim 1/320$
F. $\sim 1/256$
G. $\sim 1/255$
H. $\sim 1/128$
I. $\sim 1/127$
J. $\sim 1/32$
K. $\sim 8/255$
L. $\sim 1/16$
M. $\sim 1/8$
N. $\sim 1/7$
O. $\sim 1/4$
P. $\sim 1/2$
Q. $\sim 2/3$
R. ~ 1
S. $\sim 3/2$
T. ~ 2
U. ~ 3
V. ~ 4
W. ~ 7
X. ~ 8

14. **Algorithm design.**

(a) Here is an elegant solution:

- For each string s, form its L circular suffixes and suffix sort them (using LSD radix sort). Use the lexicographically first sorted suffix as its *fingerprint*. Two strings are cyclic rotations of one another if and only if they have the same fingerprint.
- Sort the N fingerprints (using LSD radix sort) and check adjacent fingerprints for equality. If any two are equal, then output *yes*; otherwise output *no*.

(b) The order of growth of the running time is NL^2.

- Explicitly forming the L circular suffixes of a string takes L^2 time and space. Sorting the L suffixes (each of length L) them takes L^2 time using LSD radix sort. Doing this for each string takes a total of NL^2 time.
- Sorting the N fingerprints (each of length L) takes NL time using LSD radix sort. Checking for adjacent entries that are equal also takes NL time.

The running time can be improved to NL in the worst case by implicitly forming the L circular suffixes of a string and using a linear-time suffix sorting algorithm to compute its fingerprint.

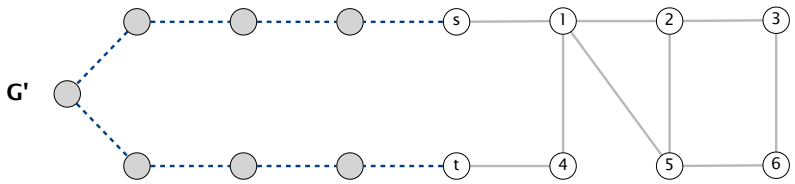
Here is an alternate NL^2 time solution:

- Explicitly (or implicitly) form the L circular suffixes of each of the N strings and put all NL of them into an array.
- Sort the NL strings using LSD radix sort.
- Check for adjacent entries that both are equal *and are circular suffixes of different original strings*. The latter check is necessary if one of the original strings happens to be a nontrivial cyclic rotation of itself, such as `stackstack`.

15. **Reductions.**

(a) We need to create a graph G' such that its longest cycle corresponds to a longest s-t path in G. The intuition is that adding an edge between s and t turns any s-t path of length k in G into a cycle of length $k+1$ in G'. This doesn't quite work because there might be a cycle in G that is longer than the length of the longest s-t path.

Instead of adding an edge between s and t, we add a new path (with new vertices) between s and t of length V. Now, s-t paths in G of length $k \geq 1$ are in 1-1 correspondence with cycles in G' of length $k+V$. Thus, finding the longest cycle in G' provides the longest s-t path in G.



(b) (i) and (iii)