COS 226	Algorithms and Data Structures	Fall 2017

Final Exam Solutions

1. **Initialization.**

   Don't forget to do this.

2. **Memory.**

   $\sim 48n$ bytes

   Each `Node` *object requires 48 bytes: object overhead (16 bytes), 3 references (24 bytes),* `char` *(2 bytes),* `int` *(4 bytes), padding (2 bytes).*

3. **Running time.**

   E D D D D E

4. **String sorts.**

   A Original input
   C MSD radix sort after the second call to key-indexed counting
   D 3-way radix quicksort after the first partitioning step
   C MSD radix sort after the first call to key-indexed counting
   B LSD radix sort after 1 pass
   D 3-way radix quicksort after the second partitioning step
   E Sorted

5. **Depth-first search.**

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

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

   (c) Explanation 1: There cannot be a topological order because of the directed cycle $5 \to 3 \to 9 \to 5$.

   Explanation 2: If G were a DAG, then we know that the reverse postorder would be a topological order. However, the reverse of the postorder from (b) is not a topological order (e.g., because 5 appears before 9 in the reverse postorder but $9 \to 5$ is an edge).

6. **Breadth-first search.**

   0 4 8 5 9 2 3 1 7 6

7. **Maximum flow.**

   (a) $50 = 9 + 3 + 38$

   (b) $78 = 29 + 12 + 37$

   (c) 5

   (d) $A \to B \to C \to H \to I \to D \to J$

   (e) The unique mincut is $\{A, B, C, F, G\}$.

8. **LZW compression.**

   (a) C A A C A B C A B A

   (b)

i	*codeword*
81	CA
82	AA
83	AC
84	CAB
85	BC
86	CABA

9. **Ternary search tries.**

   TIGER, TO, TOO, TRIE

10. **Knuth–Morris–Pratt substring search.**

	0	1	2	3	4	5	6	7
A	0	0	3	0	0	6	0	0
B	0	0	0	0	0	0	0	8
C	1	2	2	4	5	2	7	5
s	C	C	A	C	C	A	C	B

11. **Programming assignments.**

   (a)

   - ■ There is exactly one vertex of outdegree 0.
   - □ There is exactly one vertex of indegree 0.
   - ■ There are no directed cycles.
   - □ There is a directed path between every pair of vertices.
   - ■ There are $V - 1$ edges, where V is the number of vertices.
   - □ There are $E - 1$ vertices, where E is the number of edges.

   (b) $W H$

   (c)

   - ■ A achieves a better compression ratio than B.
   - □ C achieves a better compression ratio than A.
   - □ E achieves a better compression ratio than A.
   - ■ D achieves the best compression ratio among A–E.

   (d) Percolation, WordNet, SeamCarving

12. **Properties of minimum spanning trees.**

   A C C A C

13. **Properties of shortest paths.**

   B C A D C

14. **Regular expressions.**

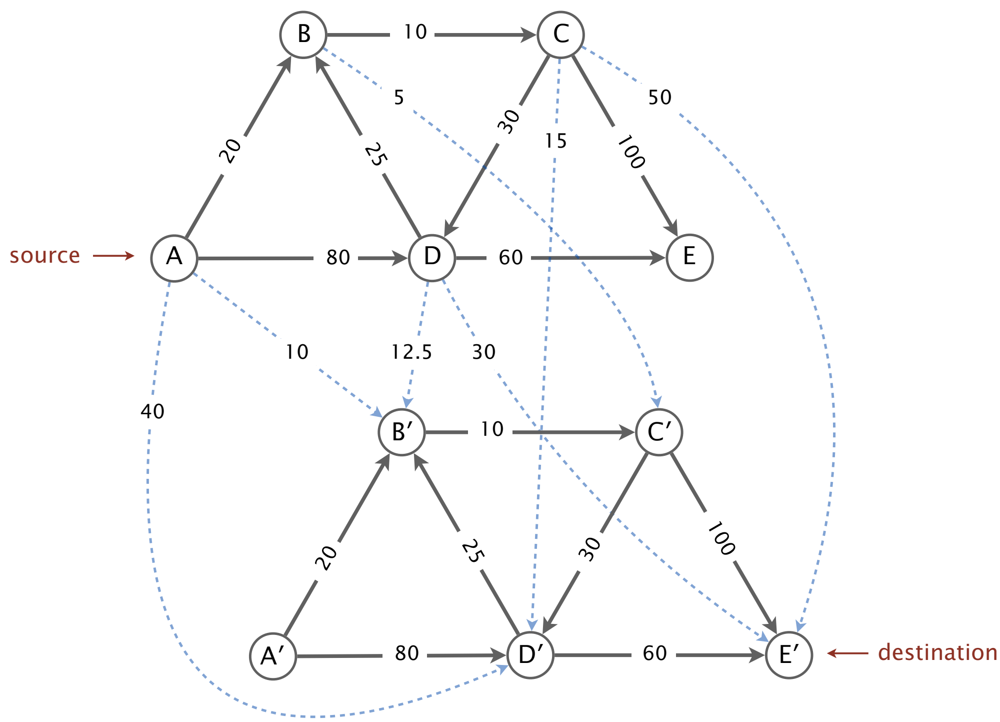
   (a) (A* | (AB*A)+)

   (b) 1 2 3 6 7 8 11 12

15. **Shortest discount path.**

Use the graph-doubling trick (ala SHORTEST-PRINCETON-PATH from the Spring 2015 Final) and create a digraph G' with $2V$ vertices and $3E$ edges as follows:

- For each vertex v in G: create two vertices v and v'.
- For each edge $v \to w$ in G: create the three edges $v \to w$, $v' \to w'$, and $v \to w'$. The weight of $v \to w$ and $v' \to w'$ equals the weight of e; the weight of $v \to w'$ is one-half that weight.



A shortest path from s to t' corresponds to a shortest discount path: the one edge in the path going from the first copy of the digraph to the second copy corresponds to the discounted edge.

16. **Substring of a circular string.**

Let u denote the string containing the first $m + n$ characters of the (infinite) circular string t. Do a substring search of the query string s in the text string u. If we use Knuth–Morris–Pratt, the overall running time will be proportional to $m + n$ in the worst case (m to build the DFA and $m + n$ to simulate it on string u).

Here are two examples, one with $m < n$ and one with $m > n$:

- s = ABBA, t = BABBBBBABBBBBAB, $m = 4$, $n = 15$. Search for the query string s = ABBA in the text string u = BABBBBBABBBBB<u>ABBA</u>BB.
- s = BBAABBAABBAABB, t = ABBA, $m = 14$, $n = 4$. Search for the query string s = BBAABBAABBAABB in the text string u = A<u>BBAABBAABBAABB</u>AA.

Note 1: Two copies of t is not enough when $m >> n$; $\lceil m/n \rceil$ copies of t is not enough when $m < n$.

Note 2: It simplest to form the string u explicitly, but you can also run Knuth–Morris–Pratt on u implicitly by building the DFA for s and simulating it on t, wrapping around to the beginning of t after you reach the end of t. In this case, you need to be careful about when to stop the simulation if no match is found: $m + n$ DFA transitions suffice.