COS 226 **Algorithms and Data Structures** **Spring 2014**

Final Exam Solutions

1. **Analysis of algorithms. (8 points)**

   (a) 20,000 seconds
   The running time is cubic.

   (b) $\sim 48N$ bytes.
   Each `Node` *object uses 48 bytes: 16 (object overhead) + 8 (inner class) + 8 (reference to* `Value`*) + 16 (references to two* `Node` *objects).*

   (c) B G L D F

2. **Graph search.**

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

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

3. **Maximum flow.**

   (a) 26

   (b) A-G-B-H-C-D-E-J

   (c) 30

   (d) A B F G H I

   (e) $B \to C, I \to D,$ or $I \to J$
   Decreasing the weight of any edge in the mincut by one will decrease the value of the mincut by one, which equals the value of the maxflow.

4. **Shortest paths.**

   (a) 5 0 6 7

   (b) 3

   (c)

v	`distTo[]`	`edgeTo[]`
0		
1		
2	`49.0`	$3 \to 2$
3		
4	`39.0`	$3 \to 4$
5		
6		
7		
8	`50.0`	$3 \to 8$
9	`54.0`	$3 \to 9$

5. **String sorting algorithms.**

   0 2 3 1 3 1 1 2 4

6. **Ternary search tries.**

   A GCA TA TCTT TGT

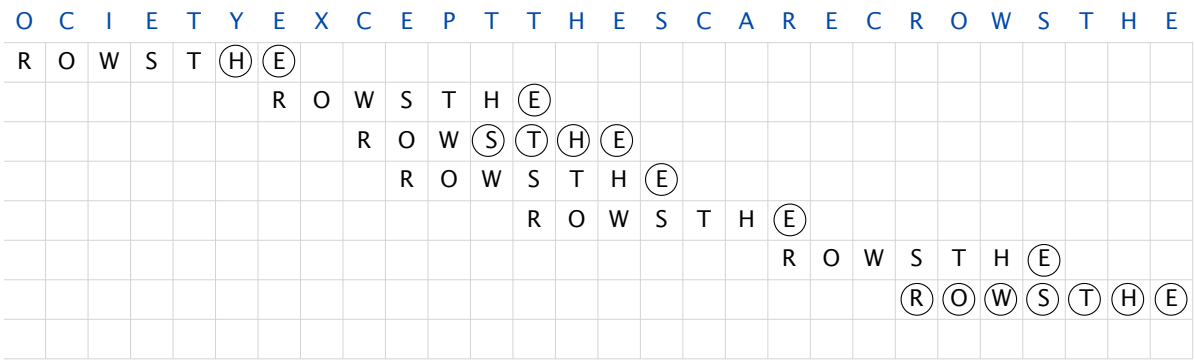
   The node labeled ? must contain the character G: it is the left link of its parent (so it must be less than T) and it is the right-left child of its grandparent (so it must be greater than C).

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

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

8. **Boyer-Moore substring search.**

   (a)



   (b) ETYEXCE

   More generally, any string that ends in YEXCE.

9. **Regular expressions.**

   (a) $(A* \mid (AB*A)*)$

   (b) 1 2 3 6 7 8 11 12

   (c) Remove the ϵ-transition edge from 4 to 10.

10. **LZW compression.**

    C A B B A B C C A A C A A A B

11. **Burrows-Wheeler transform.**

(a) 4
D B C A C A A B

(b) C D A B D D C C

12. **Problem identification.**

C *This is the* Hamilton-Path *problem, which is NP-complete. So, unless $P = NP$, there does not exist a polynomial-time algorithm for* Hamilton-Path*, but this is unknown.*

A *There is a directed path between every pair of vertices if and only if the digraph has a single strong component. You can determine this in linear time by either of the following two algorithms:*

- *Run Kosaraju-Sharir to compute the number of strong components.*
- *Pick an arbitrary vertex s; check whether every vertex is reachable from s (via DFS); and check whether s is reachable from every vertex (via DFS in the reverse digraph).*

A *You did this on the Wordnet assignment.*

C *The running time of the best-known algorithm for computing the maxflow or mincut is around EV but the asymptotic complexity remains an open research question.*

A *This can be solved by a modified version of breadth-first search, where the vertices discovered by following orange edges are put at the back of the queue (as usual) and the vertices discovered by following black edges are put at the front of the queue.*

A *Sort the integers in linear time, e.g., using LSD radix sort. Then, scan a pointer i from left to right and a pointer j from right to left. Compare $a_i + a_j$ with 0: if equal, then done; if less, then increment i; if greater, then decrement j.*

A *Key-indexed counting accomplishes this.*

A *Modify the Knuth-Morris-Pratt DFA so that the accept state has edges leaving it.*

A *If the input has exactly 10000 bits and it starts with a 1, encode it by removing the leading 1; otherwise prepend 10000 1s to the input.*

13. **Reductions.**

(a) i. Given s, we construct a string s' of length $N+1$ as follows:

- Replace each occurrence of A with 1
- Replace each occurrence of B with 2
- Append 0 to the end of the resulting string

The key to understanding the reduction is observing that the i^{th} suffix of s is less than the j^{th} suffix of s if and only if the i^{th} circular suffix of s' is less than the j^{th} circular suffix of s'. This follows because the sentinel character 0 is less than any other character (so there are no wraparound effects when comparing circular suffixes).

ii. If s = ABAAB, we would construct the string s' = 121120.

iii. To complete the reduction, we set `sa[i]` = `csa[i-1]` for `i` = 0 to $N-1$, as below.

i	s[i]	s'[i]	csa[i]	sa[i]
0	A	1	5 (012112)	2 (AAB)
1	B	2	2 (112012)	3 (AB)
2	A	1	3 (120121)	0 (ABAAB)
3	A	1	0 (121120)	4 (B)
4	B	2	4 (201211)	1 (BAAB)
5		0	1 (211201)	

(b) i. Given s, we construct a string s' of length $2N$ as follows:

- Replace each occurrence of 0 with AA
- Replace each occurrence of 1 with AB
- Replace each occurrence of 2 with BA
- Replace each occurrence of 3 with BB

The key to understanding the reduction is observing that the i^{th} circular suffix of s is less than the j^{th} circular suffix of s if and only if the $(2i)^{th}$ circular suffix of s' is less than the $(2j)^{th}$ circular suffix of s'.

ii. If $s = 03122$, we would construct the string $s' =$ AABBABBABA.

iii. To complete the reduction, we discard the entries in `csa'[]` that correspond to odd circular suffixes (and scale the entries that correspond to even circular suffixes by 2), as below.

i	s[i]	s'[i]	csa'[i]	csa[i]
0	0	A	9 (AAABBABBAB)	0 (03122)
1	3	A	0 (AABBABBABA)	2 (12203)
2	1	B	7 (ABAAABBABB)	4 (20312)
3	2	B	4 (ABBABAAABB)	3 (22031)
4	2	A	1 (ABBABBABAA)	1 (31220)
5		B	8 (BAAABBABBA)	
6		B	6 (BABAAABBAB)	
7		A	3 (BABBABAAAB)	
8		B	5 (BBABAAABBA)	
9		A	2 (BBABBABAAA)	

Alternatively, we can construct the string s' using some other prefix-free code, such as:

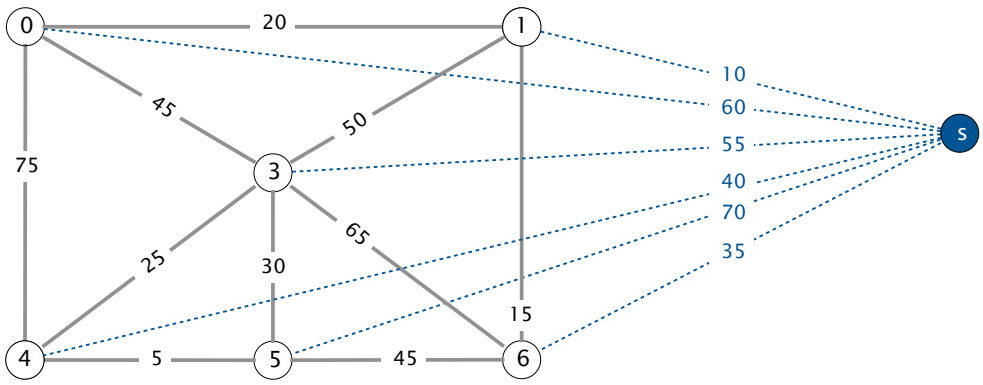
- Replace each occurrence of 0 with A
- Replace each occurrence of 1 with AB
- Replace each occurrence of 2 with ABB
- Replace each occurrence of 3 with ABBB

However, this construction makes it a bit more awkward to compute `csa[]` from `csa'[]`.

14. **Algorithm design.**

Create an edge-weighted graph with $N+1$ vertices (one for each dorm room plus an artificial source vertex s).

- Include an edge between i and j with weight c_{ij} (to represent potential fiber connections).
- Include an edge between s and i with cost w_i (to represent potential routers).



Now, if the MST contains any edge of the form s-i, then we install a router in dorm room i; if the MST contains any edge i-j not of this form, then we build a fiber connection between dorm rooms i and j.