COS 226 **Algorithms and Data Structures** **Spring 2012**

Final Solutions

1. **Analysis of algorithms.**

   (a) $f(N) = \frac{1}{100} N^{1/2}$

   (b) $\sim 24N + 64M$ bytes

   - Object overhead (16 bytes)
   - inner class (8 bytes)
   - four references (32 bytes)
   - char (2 bytes)
   - padding (6 bytes)
   - `Integer` objects ($24N$ bytes)

   (c) B E H C G A

2. **Miscellaneous.**

   C A D A C/D B A A C A

   We accepted either C or D for finding a longest path from s to t in an edge-weighted digraph. If path is required to be simple (as usual), then it can be solved in exponential time. If the path does not need to be simple, then the longest path can be made arbitrarily long by repeatedly going around a directed cycle.

3. **Graph search.**

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

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

4. **Minimum spanning trees.**

   (a) 1 2 3 4 5 7 9 14

   (b) 4 5 1 7 3 2 9 14

5. **Shortest paths.**

   (a) 5 2 4 8 7

   (b)

v	distTo[]	edgeTo[]
0	19.0	$6 \to 0$
1	10.0	$6 \to 1$
2	1.0	$5 \to 2$
3	24.0	$6 \to 3$
4	3.0	$5 \to 4$
5	0.0	*null*
6	7.0	$4 \to 6$
7	6.0	$4 \to 7$
8	4.0	$5 \to 8$
9	11.0	$6 \to 9$
10	11.0	$6 \to 10$

6. **String sorting.**

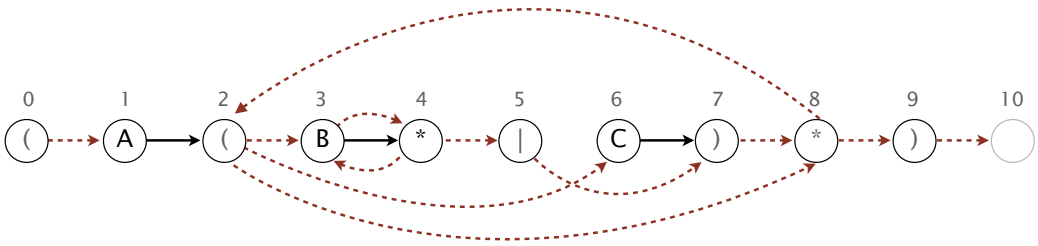
   `rabid cable table fable sable cache ...  hedge wedge ledge media medic`

7. **Substring search.**

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

8. **Regular expressions.**

   $2 \to 6, \quad 2 \to 8, \quad 3 \to 4, \quad 4 \to 3, \quad 4 \to 5, \quad 5 \to 7, \quad 8 \to 2, \quad 8 \to 9$



9. **Ternary search tries.**

   (a) 10

   `COLONIAL` and `COTTAGE` are at depth 10; `CLOISTER`, `TERRACE`, and `TOWER` are at depth 9; `TIGER` is at depth 8; `CHARTER` is at depth 7; `CANNON` and `QUAD` are at depth 5; `CAP` and `IVY` are at depth 3.

   (b) 8

   Since `COLONIAL` and `CLOISTER` each contain 8 characters, at least one of them will end up at depth 8 or higher.

   The following order leaves all strings at depth 8 or lower, which is the best possible: `COLONIAL`, `CLOISTER`, `COTTAGE`, `TERRACE`, `CHARTER`, `CANNON`, `TIGER`, `TOWER`, `QUAD`, `CAP`, and `IVY`. `CLOISTER`, `CHARTER`, and `CANNON` are at depth 8; `COLONIAL`, `COTTAGE`, `TERRACE`, and `TOWER` are at depth 7; `CAP` and `TIGER` are at depth 6; `IVY` and `QUAD` are at depth 5.

10. **Burrows-Wheeler transform.**

    (a) `0`
    `B C C A B C C B`

    (b) `C B A C B A C B D`

11. **Maximum flow.**

    (a) 37

    (b) There are four possible augmenting paths (all with bottleneck capacity equal to 1).

    - $s \to 3 \to 2 \to 5 \to t$
    - $s \to 3 \to 2 \to 5 \to 4 \to t$
    - $s \to 3 \to 6 \to 2 \to 5 \to t$
    - $s \to 3 \to 6 \to 2 \to 5 \to 4 \to t$

    (c) 38

    (d) There are two possible minimum cuts

    - $\{s, 2, 3, 6\}$
    - $\{s, 1, 2, 3, 6\}$

    (e) $38 = 9 + 15 + 14$

12. **Algorithm design.**

(a) Form the digraph G' consisting only of edges of weight greater than or equal to T. Use BFS or DFS to determine whether there is any path from s to t in G'. Alternatively, run BFS or DFS as usual in G, but ignore edges with weight strictly less than T.

(b) Use a version of binary search to find the largest threshold value T such that there exists a path from s to t of bottleneck capacity T but there is not one of capacity strictly more than T. To accomplish this, first sort the edges in increasing order of weight. $w_1, w_2, \ldots, w_E$. If there is no path from s to t or if there is a bottleneck path of capacity w_E, then we are done. Otherwise, initialize $lo = 1$ and $hi = E$, maintaining the invariant that there is a path from s to t of bottleneck capacity w_{lo} but not one of bottleneck capacity w_{hi}.

- Set $mid = (lo + hi)/2$.
- Using the subroutine from (a), determine whether this is a path from s to t of bottleneck capacity greater than or equal to w_{mid}.
- If yes, set $lo = mid$ and repeat; if no, set $hi = mid$ and repeat.

13. **Reductions.**

(a) Form the $2N$ integers as follows:

$$b_0 + 6M, \quad b_1 + 6M, \quad b_2 + 6M, \quad b_3 + 6M, \quad b_4 + 6M$$
$$-8b_0 - 12M, \quad -8b_1 - 12M, \quad -8b_2 - 12M, \quad -8b_3 - 12M, \quad -8b_4 - 12M$$

where M equal to 1 + maximum absolute value of any integer in b. Observe that all of the terms with $6M$ are strictly positive and all of the terms with $12M$ are strictly negative.

To see why it works, we must show that there exists i, j, and k such that $b_i + b_k = 8b_k$ if and only if there exists i, j, and k such that $a_i + a_j + a_k = 0$.

- Suppose that there exists i, j, and k such that $b_i + b_j = 8b_k$. Then, the entries $a_{i'} = b_i + 6M$, $a_{j'} = b_j + 6M$, and $a_{k'} = -8b_k - 12M$ satisfy $a_{i'} + a_{j'} + a_{k'} = 0$ because the $6M$ and $12M$ terms cancel out.
- Suppose that there exists i, j, and k such that $a_i + a_j + a_k = 0$. The terms must be of the form $b_{i'} + 6M$, $b_{j'} + 6M$, and $-8b_{k'} - 12M$: if no entry contains a term with $12M$, then the sum will be strictly positive; if more than one entry contains a term with $12M$, then the sum will be strictly negative. Thus, $b_{i'} + b_{j'} = 8b_{k'}$ because the $6M$ and $12M$ terms cancel out.

Remark: If you use $+M$ and $-2M$ (instead of $+6M$ and $-12M$), then the reductions fails because the $-2M$ terms may not be negative. The array `b[] = { -2, 0, -4, 9 }` provides a counterexample since it has no solution but the resulting array `a[] = {8, 10, 6, 19, -4, -20, 12, -92 }` has the solution $8 + -20 + 12 = 0$.

(b) I and III only.