Department of Electrical Engineering
Iris Hui-Ru Jiang

October 2, 2017

Algorithms, Fall 2017

Sample Solutions to Homework #1

1. (10)
   This is true. Indeed, consider such a pair (m, w) and consider a perfect matching containing pairs (m, w') and (m', w), and, hence, not (m, w). Then since m and w each rank the other first, they each prefer the other to their partners in this matching, and so this matching cannot be stable.

2. (25)
   For each schedule, we have to choose a *stopping port*: the port in which the ship will spend the rest of the month. Implicitly, these stopping port will define truncations of the schedules. We will say that an assignment of ships to stopping ports is *acceptable* if the resulting truncations satisfy the conditions of the problem — specifically, condition (*). (Note that because of condition (*), each ship must have a distinct stopping port in any acceptable assignment.)
   We set up a stable marriage problem involving ships and ports. Each ship ranks each port in chronological order of its visits to them. Each port ranks each ship in reverse chronological order of their visits to it. Now we simply have to show:

   (1) *A stable matching between ships and ports defines an acceptable assignment of stopping ports.*

   Proof. If the assignment is not acceptable, then it violates condition (*). That is, some ship S_i passes through port P_k after ship S_j has already stopped there. But in this case, under our preference relation above, ship S_i "prefers' P_k to its actual stopping port, and port P_k "prefers" ship S_i to ship S_j. This contradicts the assumption that we chose a stable matching between ships and ports.

3. (20)
   We know from the text that polynomials (i.e. a sum of terms where n is raised to fixed powers, even if they are not integers) grow slower than exponentials. Thus, we will consider f_1, f_2, f_3, and f_6 as a group, and then put f_4 and f_5 after them.
   For polynomials f_i and f_j, we know that f_i and f_j can be ordered by comparing the highest exponent on any term in f_i to the highest exponent on any term in f_j. Thus, we can put f_2 before f_3 before f_1. Now, where to insert f_6? It grows faster than n^2, and from the text we know that logarithms grow slower than polynomials, so f_6 grows slower than n^c for any $c > 2$. Thus we can insert f_6 in this order between f_3 and f_1.
   Finally come f_4 and f_5. We know that exponentials can be ordered by their bases, so we put f_4 before f_5.

4. (25)
   Suppose that to obtain n words, we need L lines (most of which will get repeated many times, as described above). We write the script as follows:

   ```
   line 1 = <text of line 1 here>
   line 2 = <text of line 2 here>
   ```

```
...
line L = <text of line L here>
For i = 1, 2, ..., L
   For j = 1, 2, ..., i
     Sing lines j through 1
   Endfor
Endfor
```

Now, the nested `For` loops have length bounded by a constant c_1, so the real space in the script is consumed by the text of the lines. Each of these lines in the script has length at most c_2 (where c_2 is the maximum line length c plus the space to write the variable assignment).
So in total, the space required by the number of words this produces when sung. n is at least $1+2+...+L = \frac{1}{2}L(L-1)$; hence, $\frac{1}{2}(L-1)^2 \leq n$, and so $L \leq 1+\sqrt{2n}$. Plugging this into our bound on the length of the script. we have $f(n) = S \leq c_1 + c_2\sqrt{2n} = O(\sqrt{n})$.

5. (20)

   (a) From pseudo code, we can know that the time complexity of this algorithm is $O(N^4)$, where N is the input size. Since the input size is 10 times larger than the previous one, the running time should be 10^4 larger, hence the estimated running time for $N = 10000$ is 10^7 seconds.

   (b) Suppose the formula that estimate the running time T as a function of N is as follows:

   $$T = cN^4,$$

   where c is the coefficient. Given the case: $N = 1000 \rightarrow T = 1000$, we have:

   $$c = 10^{-9}(sec).$$