COMS21103 - Coursework 1

Ross Gardiner

November 26, 2015

2 a)

$$\mathrm{P}[i, x] = \begin{cases} 0 & \text{if } i = 0 \\ max(P[i-1, s] - m, P[i-1, u]) + U[i] & \text{if } x = u \\ max(P[i-1, s], P[i-1, u] - m) + S[i] & \text{if } x = s \end{cases}$$

2 b) The correctness of this formula can be shown by induction. Informally, after 0 days, the profit must be 0. After > 0 days, the profit will be the (max profit from the current day)+(the profit from the previous days).

The scenario in which a greedy algorithm would fail is where it has to choose whether to switch or stick without knowing the future impact of that decision.

Because the recursive algorithm relies on the pre-calculated maximum prior profit, it will only switch stock if the switch increases takings by at least the cost of switching plus the takings from the other stock type. As a result, it will never end up in a situation where it switches stock inefficiently.

2 f) TODO

2 a) Magic-Tournament$[magician]$ = Maximum $\left[\mathrm{M}[magician, true],\ \mathrm{M}[magician, false]\right]$
(where *magician* is the head magician)

Let:
$m \equiv$ root magician of some subtree
$s \equiv$ is the root magician included?

Using Haskell-ish notation:

$$\mathrm{M}[m, s] = \begin{cases} (\textsc{Fold}\ (+)\ (\textsc{Map}\ (a \mapsto \mathrm{M}[a, false])\ m.apprentices)) + m.ability & \text{if } s \\ \textsc{Fold}\ (+)\ (\textsc{Map}\ (a \mapsto (\textsc{Maximum}\ \left[\mathrm{M}[a, true],\ \mathrm{M}[a, false]\right]))\ m.apprentices) & \text{if } \neg s \end{cases}$$

2 b) The algorithm explores every possibility. Each magician in the tree is at the root of his own tree—possibly empty. M can be called such that it either includes the root magician or excludes the root magician. In the former case, all of the apprentices must be excluded. In the latter case, the algorithm finds the maximum result from either including or excluding each apprentice.

At the root of the tree, we try calling M both including and excluding the head magician, and pick the highest result.

2 d) TODO

4 a) Let:
S$[d, r]$ returns a tuple of:

- The most number of sandwiches legally deliverable by the end of day d (where d : int), given whether the last day is a rest day (indicated by r, where r : bool).
- The length of the contiguous run of non-rest days up to and including day d.

N.B. The notation $(\_, \_, ..., \_)_n$ retrieves the nth item of a tuple. For example, $(52, 74)_2 = 74$.

$$\mathrm{S}[d, r] = \begin{cases} (0, 0) & \text{if } d = 0 \\ (\mathrm{S}[d-1, false]_1, 0) & \text{if } r \\ ((\textsc{Min} \big[& \\ \quad M[S[d-1, true]_2 + 1], B[d]\big]) & \\ \quad + S[d-1, true]_1, & \\ S[d-1, true]_2 + 1) & \text{if } \neg\, r \wedge \\ & (\textsc{Min} \left[M[S[d-1, true]_2 + 1], B[d]\right]) + S[d-1, true]_1 > \\ & (\textsc{Min} \left[M[1], B[d]\right]) + S[d-1, false]_1 \\ (\textsc{Min} \left[M[1], B[d]\right] + S[d-1, false]_1, 1) & \text{otherwise} \end{cases}$$

4 b) TODO

4 c) TODO

4 f) TODO

4 h) TODO