

1 Problem 6.2

Let $b[i]$ be the minimum total penalty for stopping at hotel a_i , $1 \leq i \leq n$. We have

$$b[i] = \min_{1 \leq j < i} \{b[j] + (200 - (a[i] - a[j]))^2\}.$$

Also record the value j which yields $\min_{1 \leq j < i} \{b[j] + (200 - (a[i] - a[j]))^2\}$ and set $b[i].\text{prev} = b[j]$. Backtrack from $b[n]$ to get the sequence of hotels to stop by.

2 Problem 6.3

Let $S[i]$ be the maximum total profit we get from building some restaurants in $\{m_1, m_2, \dots, m_i\}$. Consider 2 cases:

- (1) If restaurant m_i should not be built, then $S[i] = S[i - 1]$.
- (2) If restaurant m_i should be built, then let c_i be the maximum index j which yields $m_i - m_j \geq k$. We then have $S[i] = p_i + S[c_i]$.

Therefore in general,

$$S_i = \max\{S[i - 1], p_i + S[c_i]\}.$$

To get the sequence of restaurants, keep an array $R[n]$ such that $R[i] = 1$ if restaurant i is built in the optimal solution, and $R[i] = 0$ otherwise. When we calculate $S[i]$, if the max falls to case (1), $R[i] = 0$. Else, $R[i] = 1$. Output all the $R[i]$ s that are 1.

3 Problem 6.4

Consider an array $S[n]$ where $S[i] = \text{true}$ if the substring $s_1 s_2 \dots s_i$ is a valid string, and **false** otherwise. We have $S[1] = \text{dict}(s[1])$ and

$$\begin{aligned} S[i] = & (S[1] \ \&\& \ \text{dict}(s[2 \dots i]) \\ & || (S[2] \ \&\& \ \text{dict}(s[3 \dots i]) \\ & || \dots \\ & || (S[i - 1] \ \&\& \ \text{dict}(s[i \dots i]))), \end{aligned}$$

where $s[j..i]$ is $s_j s_{j+1} \dots s_i$.

4 Problem 6.6

Let the input string be $x_1 x_2 \dots x_n$.

Let $Z = \{a, b, c\}$ and let $T[i, j] \subset Z$ be the set of the possible values that the product $x_i x_{i+1} \dots x_j$ can yield with all possible parenthesizations.

We see that $T[i, i] = x_i$ for all $1 \leq i \leq n$. We need to compute $T[1, n]$.

Define $A \times B$ as $\{a \cdot b | a \in A, b \in B\}$.

We note that $T[i, i + 1] = T[i, i] \cup T[i + 1, i + 1]$ and $T[i, i + 2] = (T[i, i] \times T[i + 1, i + 2]) \cup (T[i, i + 1] \times T[i + 2, i + 2])$ (to put it another way, abc can be written as $(a)(bc)$ or $(ab)(c)$).

We therefore see that we already have

$$T[1, 1], T[2, 2], T[3, 3], \dots,$$

from which we can calculate

$$T[1, 2], T[2, 3], T[3, 4], \dots,$$

from which we can calculate

$$T[1, 3], T[2, 4], T[3, 5], \dots$$

and eventually we can expand to $T[1, n]$, which is what we need to find.

In other words,

$$T[i, i + s] = \bigcup_{i \leq k < i+s} (T[i, k] \times T[k + 1, i + s]).$$

The algorithm is as follows:

```

for i = 1 to n: T[i,i] = x[i].
for s = 1 to n-1:
  for i = 1 to n - s:
    T[i, i + s] = empty
    for k = 1 to i + s - 1:
      T[i, i + s] = T[i, i + s] UNION (T[i, k] * T[k+1,s])
If a is in T[1,n] return true. Else, return false.

```

5 Problem 6.7

Let the input string be $x_1x_2 \dots x_n$. Let $T[i, j]$ be the length of the longest palindromic subsequence in $x[i..j]$. We have

$$\begin{cases} T[i, i] = 1 \\ T[i, i + 1] = 2 \text{ if } x[i] = x[i + 1] \text{ and } 0 \text{ if } x[i] \neq x[i + 1] \\ T[i, j] = T[i + 1, j - 1] + 2 \text{ if } x[i] = x[j] \text{ and } \max\{T[i + 1, j], T[i, j - 1]\} \text{ else} \end{cases}$$

6 Problem 6.8

Let $E[i, j]$ be the length of the largest common substring of $x_1x_2 \dots x_i$ and $y_1y_2 \dots y_j$ such that $x_i = y_j$. We see that $E[1, j] = 1$ if $x_1 = y_j$ and 0 otherwise. Similarly, $E[j, 1] = 1$ if $y_1 = x_j$ and 0 otherwise. In general, we have

$$E[i, j] = \begin{cases} E[i - 1, j - 1] + 1 \text{ if } x_i = y_j \\ 0 \text{ if } x_i \neq y_j \end{cases}$$

7 Problem 6.9

Let the input string be $x[0..n - 1]$ and the input breakpoint array be $y[1..m]$. Convert y to $y[0..m + 1]$ and let $y[0] = -1, y[m + 1] = n - 1$.

Let $M(i, j)$ be

$$\begin{cases} M(i, i) = 0, \forall i : 0 \leq i \leq m + 1 \\ M(i, i + 1) = 0, \forall i : 0 \leq i \leq m + 1 \\ M(i, j) = (y[j] - y[i]) + \min_{l: i < l < j} \{M(i, l) + M(l, j)\} \end{cases}$$

8 Problem 6.10

Let $E[i, j]$ be the probability of obtaining exactly i heads when j coins c_1, c_2, \dots, c_j with head-probability p_1, p_2, \dots, p_j are tossed. We have $E[0, j] = (1 - p_1)(1 - p_2) \dots (1 - p_j)$ and

$$E[i, j] = 1 - \sum_{k=1}^i E[i - k, j].$$

9 Problem 6.11

Let $E[i, j]$ be the longest common subsequence of $x_1x_2 \dots x_j$ and $y_1y_2 \dots y_j$.

We have $E[1, j] = 1$ if $x_1 = y_j$, for all $1 \leq j \leq m$. Similarly, $E[j, 1] = 1$ if $x_j = y_1$, for all $1 \leq j \leq n$.

In general we have

$$E[i, j] = \begin{cases} E[i-1, j-1] + 1 & \text{if } x_i = y_j \\ \max\{E[i-1, j] + E[i, j-1]\} & \text{if } x_i \neq y_j. \end{cases}$$

10 Problem 6.12

Let $d[i, j]$ be the distance between point i and j . We have

$$\begin{cases} A[i, i] = 0 \\ A[i, i+1] = 0 \\ A[i, i+2] = 0 \\ A[i, j] = \min_{i < k < j} \{A[i, k] + A[k, j] + d[i, k] + d[k, j]\} \end{cases}$$

11 Problem 6.13

Let $E[i, j]$ be the maximum value the first player can have by picking cards from the set of cards s_i, s_{i+1}, \dots, s_j .

We see that $E[i, j] = 0$ if $i \leq j$. In general,

$$E[i, j] = \max\{v_i + \min\{E[i+2, j] - v_{i+1}, E[i+1, j-1] - v_j\}, \\ v_j + \min\{E[i+1, j-2] - v_{j-1}, E[i+2, j-1] - v_{i+1}\}\}.$$