Algorithm 2020 Spring: Assignment Week 6

Due on Monday, April 6, 2020

李胜锐 2017012066

Question 1

p222 15.3-4

For 3 matrixes: $A \times B \times C = (1 * 2) \times (2 * 30) \times (30 * 10)$
Solution of Capulet is: $(A \times B) \times C, n = 1 * 2 * 30 + 1 * 30 * 20 = 660$
The best solution is: $A \times (B \times C), n = 2 * 30 * 10 + 1 * 2 * 10 = 620$
Obviously, solution of Capulet can't be the best solution.

Question 2

Seam Carving

考虑从第一行开始依次删除像素。对于每一行，如果确定了删除像素的位置，那么下一行可以删除的位置为左下、正下、右下（不考虑在边界的特殊情况），共有三种选择。因此，不考虑边界，总的数量为:

$$T(n) = O(3^n)$$

考虑边界:

$$T(n) = \Omega(2^n)$$

显然是指数函数.

Question 3

Seam Carving **算法设计**

采用动态规划算法。
从第一行开始依次向下寻找最佳路径。
每一个像素都有一个“累计破坏度”$s(i, j)$，第一行的 $s(i, j)$ 即为其自己的破坏度 $d(i, j)$。
第 i 行的某个像素 $s(i, j)$ 为其上面与它相邻的 3（2）个像素的最小累计破坏度加上它自己的破坏度，即 $min(s(i-1, j-1), s(i-1, j), s(i-1, j+1)) + d(i, j)$。
删掉一个像素后，下一次将删除的像素为该像素的前驱中累计破坏度最小的像素。依次迭代，直到像素没有前驱。

Listing 1: 伪代码

Question 3 continued on next page...

```
1
2  for row in all_rows:
3      if row == 0:  # 第一行
4          for col in all_cols:
5              s[row,col] <- d[row,col]
6      else:  # 非第一行
7          for col in all_cols:
8              # 前一行，相邻列，s代表某个像素的累计损失
9              s[row,col] <- min(s[row-1,neibor_col])+d[row,col]
10
11 min <- min(s[last_row,col])  # 得到最小损失
12 find min_col  # 最小损失的终点
13 path <- empty
14 location <- min_col
15 for row in reverse(all_rows):  # 回溯路径
16     next_location <- min(neibour(location))
17     delete location
18     location <- next_location
```

算法的时间复杂度:
为每个像素计算 $d(i,j)$:$\Theta(n^2)$
计算最小累计损失时，遍历像素:$\Theta(n^2)$
寻找最小终点、回溯路径:$\Theta(n)$
总的时间复杂度:

$$T(n) = \Theta(n^2)$$