week2

$O(n)$的理解

O(n)是指上限，只要原函数在他下面就可以！

1-3 $(logN)^3$ is $O(N)$. (5分)

T F

1-3 答案错误 ⓘ (0 分) 创建提问

2-2 下列哪个函数是$O(N)$的? (5分)

A. $(logN)^2$

B. $(NlogN)/1000$

C. $N(logN)^2$

D. $N^2/1000$

2-2 答案错误 ⓘ (0 分) 创建提问

正确答案：A

if...else...

if...else...选择最大的那个，而不是求和（像不像计组里面的求关键路径(¬ ¬)~）

求和$\sum_{i=0}^{N-1} N^2 - i = \mathrm{O}(N^3)$,

> 这种第一层循环最好理解为$sum(case_{i=1}, casei = 2, \cdots, case_{i=N-1})$，而不是$\times n$！！

2-3 下列代码 (5分)

```
if ( A > B ) {
    for ( i=0; i<N; i++ )
        for ( j=N*N; j>i; j-- )
            A += B;
}
else {
    for ( i=0; i<N*2; i++ )
        for ( j=N*2; j>i; j-- )
            A += B;
}
```

的时间复杂度是：

- A. $O(N)$
- B. $O(N^2)$
- C. $O(N^3)$
- D. $O(N^4)$

正确答案：C

while

要避坑，n在这里是固定值，不是一个变量，所以while循环的坑啊！

2-4 下列程序段的时间复杂度为（）。 (8分)

```
i = 1; k = 0; n = 100;
do{
    k = k + 10 * i;
    i = i++; i = ++i;
}while(i != n)
```

A. O(1)

B. O(n)

C. O(i)

D. O(i × n)

2-4 答案错误 ⓘ (0 分)

正确答案：A

递归和

2-5 算法分析(应用) (8分)

下面 SumPower 函数的时间复杂度为 ______ 。

```
double Power(double x, int n)
{
    double y;
    if (n > 0)
    {
        y = Power(x, n / 2);
        y *= y;
        if (n % 2)
        {
            y *= x;
        }
    }
    else
    {
        y = 1.0;
    }
    return y;
}

double SumPower(double x, int n)
{
    double y;
    if (n > 0)
    {
        y = SumPower(x, n - 1) + Power(x, n);
    }
    else
    {
```

```
        {
            y = 1.0;
        }
        return y;
    }
```

A. $O(n^2)$

B. $O(2^n)$

C. $O(\log_2 n)$

D. $O(n\log_2 n)$

E. $O(n)$

F. $O(1)$

G. $O(\sqrt{n})$

H. $O(n\sqrt{n})$

2-5 答案错误 (0 分)

> 看到递归，一般写成如下这种"递归树"形式。
>
> $T(n) = T(n-1) + logn$
>
> 最简单的求解思路：求n项和 / 求积分
>
> $T(n) = \log n + \log(n-1) + ... + \log 1$

正确答案：D

渐近界

> O 比原函数大
>
> 紧确界 常数乘数内等于
>
> 下渐近界 比原函数小
>
> 如果是Θ，那么既满足Ω也满足O. 即a=b，既有a>=b，也有a<=b成立

Laudau Operations

$O(f(n)) + \Theta(g(n)) =$ ____，选择正确的函数

(1) $\Theta(g(n))$

(2) $O(f(n)+g(n))$

(3) $\Theta(\max\{f(n), g(n)\})$

(4) $O(\max\{f(n), g(n)\})$

(5) $\Omega(g(n))$

答案：(2), (4), (5)

$O(f(n)) + \Theta(g(n))$

$\Rightarrow \exists c_1, c_2, c_3, n_0 > 0\ \forall n \geq n_0$:

$c_1 g(n) \leq O(f(n)) + \Theta(g(n)) \leq c_2 f(n) + c_3 g(n))$

week8

时间复杂度1

题目

3-1 递推方程$T(n)=T(n/4)+T(3n/4)+O(n), T(1)=O(1)$, 则$T(n)=(\ )$. (6分)

- [ ] A. $\Theta(n)$
- [x] B. $O(n^2)$
- [ ] C. $O(n\log(n))$
- [x] D. $\Theta(n\log(n))$
- [x] E. $\Omega(n)$

自己的解法

$$T(\tfrac{n}{4})+T(\tfrac{3n}{4})+O(n) \rightarrow c[\tfrac{n}{4}\log\tfrac{n}{4}+\tfrac{3n}{4}\log\tfrac{3n}{4}]+dn$$

$$=cn\log n+c[\tfrac{-n\log 4}{4}+\tfrac{3n\log 3}{4}-\tfrac{3n\log 4}{4}]+dn$$

$$=cn\log n+c[\tfrac{3n\log 3-4n\log 4}{4}]+dn$$

$$=cn\log n+[\underbrace{\underbrace{\tfrac{(\log 27-\log 256)c}{4}}_{小于0}+d}_{可>0,\ 可<0}]n$$

错误点

$T(n/4)+T(3n/4)+\mathrm{O}(n)$中，这里题目说的$\mathbf{O(n)}$，而不是$\Theta(n)$，所以$O(n)-\Theta(n)$恒正。

时间复杂度2

题目

3-2 递推方程$T(n)=T(3n/4)+T(n/4)+\Theta(n\log(n))$, $T(1)=\Theta(1)$, 则$T(n)=(\ )$。 (9分)

- [x] A. $\Theta(n^2\log(n))$
- [ ] B. $O(n^2)$ V
- [ ] C. $\Theta(n\log^2(n))$ V
- [ ] D. $\Theta(n^{1.5})$
- [x] E. $\Omega(n\log(n))$ V

错误分析

$$T(n)=2T\left(\frac{n}{4}\right)+T\left(\frac{3n}{4}\right)+\ \Theta(n\log(n))\ \text{证明}\ T(n)=\Theta(n\log^2(n))$$

证明 $T(n)=O(n\log^2(n))$

设 $T(n)\le c_1 n\log^2(n)$

$$T(n)\le c_1(\frac{n}{4}\log^2\left(\frac{n}{4}\right)+c_1\left(\frac{3n}{4}\right)\log^2\left(\frac{3n}{4}\right)+dn\log(n)$$

$$T(n)\le c_1 n\ log^2(n)+\underline{\left(c_1\frac{1}{2}\log\left(\frac{1}{4}\right)+c_1\frac{3}{2}\log\left(\frac{3}{4}\right)+d\right)}n\log(n)+c_1\left(\frac{1}{4}\log^2\left(\frac{1}{4}\right)+\frac{3}{4}\log^2\left(\frac{3}{4}\right)\right)\text{n}$$

排序

3-3 对长度为N且元素全部相等的数组执行排序，时间复杂度为$\Theta(N)$的算法有哪些? (6分)

- [ ] A. 选择排序 O(N^2)
- [x] B. 冒泡排序 O(n)
- [x] C. 插入排序 O(n)
- [ ] D. 合并排序 O(N·log N)
- [ ] E. 快速排序 O(N^2)
- [ ] F. 堆排序 O(n)
- [x] G. 希尔排序 O(N·log N)

堆排序：没有上升和下降，故是O(n)

3-4 对长度为N且元素全部相等的数组执行排序，时间复杂度为$\Theta(N\log(N))$的算法有哪些？ (6分)

- [ ] A. 选择排序
- [ ] B. 冒泡排序
- [ ] C. 插入排序
- [x] D. 合并排序
- [ ] E. 快速排序
- [ ] F. 堆排序
- [x] G. 希尔排序。

3-4 答案正确 (6 分)

3-5 对长度为N且元素全部相等的数组执行排序，时间复杂度为$\Theta(N^2)$的算法有哪些？

- [x] A. 选择排序
- [ ] B. 冒泡排序
- [ ] C. 插入排序
- [ ] D. 合并排序
- [x] E. 快速排序
- [ ] F. 堆排序
- [ ] G. 希尔排序。

3-5 答案正确 (6 分)

快排

2-2 在快速排序的一趟划分过程中，当遇到与基准数相等的元素时，如果左右指针都会停止移动，那么当所有元素都相等时，算法的时间复杂度是多少？

- A. $O(logN)$
- B. $O(N)$
- C. $O(NlogN)$ √
- D. $O(N^2)$

题解：感觉不是正规的快排，出于排序的目的，会跳过，指针会继续移动，所以最终的时间复杂度是$\mathrm{O}(n^2)$。但是题目设定的是直接停止移动，就很迷了！

2-5 从$n(>0)$个互不相同的正整数中随机选择一个数作为基准值，对所有数进行划分，把比基准值小和大的数分开成两组，则这两组的数量比的期待值是： (5分)

- A. 1 : 1 √
- B. 1 : 2
- C. 1 : 3
- D. 1 : 4

题解：

理解错误，`1:3`是选基准值分组，左：右均值为1：3；这里说的大：小期待值为1：1。

- 考题

Recursive Solution

$$c[\alpha,\beta]=\begin{cases}0 & \text{if } \alpha \text{ empty or } \beta \text{ empty},\\ c[prefix\,\alpha, prefix\,\beta]+1 & \text{if end}(\alpha)=\text{end}(\beta),\\ \max(c[prefix\,\alpha,\beta], c[\alpha, prefix\,\beta]) & \text{if end}(\alpha)\neq\text{end}(\beta).\end{cases}$$

- Keep track of $c[\alpha,\beta]$ in a table of nm entries:
  - top/down
  - bottom/up

		p	r	i	n	t	i	n	g
s									
p									
r									
i									
n									
g									
t									
i									
m									
e									

week9

NP定义

多项式（次数）不确定问题。这里都说了大小不超过n^3，已经是**确定的多项式次数**了！

1-1 考虑有n件物品的背包问题，如果没有物品的大小超过n^3，则这个问题就不再是NP难问题。 (10分)

○ T ◉ F

1-1 答案错误 ⓘ (0 分) 创建提问

week12

1-1 如果一个问题可以用动态规划算法解决，则总是可以在多项式时间内解决的。 (5分)

○ T ◉ F

1-1 答案正确 (5 分) 创建提问

> 动态规划，不是每次都是填个二维或一维表哦

< 返回 **7-2 凑零钱 (30 分)**

韩梅梅喜欢满宇宙到处逛街。现在她逛到了一家火星店里，发现这家店有个特别的规矩：你可以用任何星球的硬币付钱，但是绝不找零，当然也不能欠债。韩梅梅手边有 10^4 枚来自各个星球的硬币，需要请你帮她盘算一下，是否可能精确凑出要付的款额。

输入格式：

输入第一行给出两个正整数：N（$\le 10^4$）是硬币的总个数，M（$\le 10^2$）是韩梅梅要付的款额。第二行给出 N 枚硬币的正整数面值。数字间以空格分隔。

输出格式：

在一行中输出硬币的面值 $V_1 \le V_2 \le \cdots \le V_k$，满足条件 $V_1 + V_2 + ... + V_k = M$。数字间以 1 个空格分隔，行首尾不得有多余空格。若解不唯一，则输出最小序列。若无解，则输出 `No Solution`。

注：我们说序列{ $A[1], A[2], \cdots$ }比{ $B[1], B[2], \cdots$ } "小"，是指存在 $k \ge 1$ 使得 $A[i] = B[i]$ 对所有 $i < k$ 成立，并且 $A[k] < B[k]$。

输入样例 1：

```
8 9
5 9 8 7 2 3 4 1
```

输出样例 1：

```
1 3 5
```

输入样例 2：

```
4 8
7 2 4 3
```

输出样例 2：

```
No Solution
```

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;
const int MAXN = 1e4 + 10;
const int MAXM = 1e2 + 10;
int dp[MAXN];
int coin[MAXN];
bool choice[MAXN][MAXM];

int main()
{
    int n, m;
    cin >> n >> m;
    for (int i = 1;i<=n;i++) cin >> coin[i];
    sort(coin + 1, coin + n + 1); // 从小到大排序
    for (int i = n; i >= 1; i--) // 从大硬币到小硬币
    {
        for (int j = m; j >= coin[i]; j--)
        {
            if (dp[j] <= dp[j - coin[i]] + coin[i])
            {
                choice[i][j] = true;
                dp[j] = dp[j - coin[i]] + coin[i];
            }
        }
    }
    if (dp[m] != m){
        cout <<"No Solution"<<endl;
        return 0;
    }
    int rest = m, idx = 1;
    int flag = 0;
    while (rest > 0)
    {
        if (choice[idx][rest] == true)
        {
            if (flag) cout << " ";
            cout << coin[idx];
            rest -= coin[idx];
            flag = 1;
        }
        idx++;
    }
    cout << endl;
    return 0;
}
```

week13

选择判断题

1、最优二叉搜索树的根结点一定存放的是搜索概率最高的那个关键字。F

2、For finding an optimal binary search tree, we can use the same greedy algorithm as the one for building a Huffman tree. F

3、The time complexity to find and record the order of the optimal way to compute the multiplications of $M_1 \times M_2 \times ... \times M_n$ is $O(n^3)$ where *n* is the number of matrices. T

4、在求解最优二叉搜索树问题时，我们用到递推式$c_{ij} = min_{i \le l \le j}\{w_{ij} + c_{i,l-1} + c_{l+1,j}\}$。要通过迭代求解此式，必须用以下哪种方式填表： 正解：C

A.

```
    for i= 1 to n-1 do;
      for j= i to n do;
        for l= i to j do
```

B.

```
   for j= 1 to n-1 do;
      for i= 1 to j do;
        for l= i to j do
```

C.

```
    for k= 1 to n-1 do;
      for i= 1 to n-k do;
         set j = i+k;
         for l= i to j do
```

D.

```
    for k= 1 to n-1 do;
      for i= 1 to n do;
         set j = i+k;
         for l= i to j do
```

编程题

7-1 至多删三个字符

给定一个全部由小写英文字母组成的字符串，允许你至多删掉其中 3 个字符，结果可能有多少种不同的字符串？

输入格式：

输入在一行中给出全部由小写英文字母组成的、长度在区间 [4, 10^6] 内的字符串。

输出格式：

在一行中输出至多删掉其中 3 个字符后不同字符串的个数。

输入样例:

```
ababcc
```

输出样例:

```
25
```

提示:

删掉 0 个字符得到 "ababcc"。

删掉 1 个字符得到 "babcc", "aabcc", "abbcc", "abacc" 和 "ababc"。

删掉 2 个字符得到 "abcc", "bbcc", "bacc", "babc", "aacc", "aabc", "abbc", "abac" 和 "abab"。

删掉 3 个字符得到 "abc", "bcc", "acc", "bbc", "bac", "bab", "aac", "aab", "abb" 和 "aba"。

代码:

https://www.cnblogs.com/8023spz/p/10499968.html

https://blog.csdn.net/Mitsuha_/article/details/81123057

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAX = 1e6+5;
char s[MAX + 10];
ll dp[MAX + 10][4];
int pos[26];
int main() {
    scanf("%s",s + 1);
    dp[0][0] = 1;
    int len = strlen(s + 1);
    for(int i = 1;i <= len;i ++) {
        dp[i][0] = 1;
        int d = pos[s[i] - 'a']; // 记录上一个s[i]同样字符的位置
        pos[s[i] - 'a'] = i; // 记录这一个字符的位置
        for(int j = 1;j < 4;j ++) {
            dp[i][j] += dp[i - 1][j - 1] + dp[i - 1][j]; // 第i个字符要么删，要么留
            if(d && j - i + d >= 0) { // d不是0，且j - i + d(删的字符个数)有效
                dp[i][j] -= dp[d - 1][j - i + d]; // 到位置i删掉了 d +j - i 个字符
            }
        }
    }
    printf("%lld",dp[len][0] + dp[len][1] + dp[len][2] + dp[len][3]);
}
```

7-2 青蛙过桥_LF

一座长度为n的桥，起点的一端坐标为0，且在整数坐标i处有a[i]个石头【0<=a[i]<=4】， 一只青蛙从坐标0处开始起跳，一步可以跳的距离为1或2或3【即每一步都会落在整数点处】， 青蛙落在i处会踩着该点的所有石头，求青蛙跳出这座桥最少踩多少个石头？并且输出依次跳 过的坐标点路线，如果存在多种路线，输出字典序最小的那一条。

输入格式:

第一行整数n(<150000)，接着下一行会有n+1个由空格隔开的整数,即桥上各个坐标处石头数量。

输出格式:

第一行为踩着最少石头个数，第二行为依次跳过的坐标点【字典序最小的】。

输入样例:

在这里给出两组输入。例如:

```
10
1 2 1 3 0 3 1 2 1 1 2
100
1 2 0 4 0 1 3 4 2 2 1 3 1 4 0 3 0 1 2 3 3 2 2 0 1 0 0 0 0 1 2 1 3 4 0 3 4 4 1 0
4 1 3 1 1 2 3 4 4 4 0 2 0 1 1 1 3 1 3 2 1 2 4 1 2 1 4 1 0 0 1 2 3 0 2 4 4 0 0 4
2 0 0 2 1 3 3 3 0 0 2 0 0 1 2 4 2 2 2 4 0
```

输出样例:

在这里给出对应的输出。例如:

```
4
0 2 4 6 8
36
0 2 4 5 8 10 12 14 16 17 20 23 25 26 27 28 31 34 35 38 39 41 44 47 50 52 54 57 60 63 65 68 69 70 73 74 77 78 81 82 85 88 89 91 92 94 97 100
```

代码:

> 坑：从前往后推，要考虑0的情况
>
> 正解：从后往前推

```
#include<bits/stdc++.h>
#define DEBUG 0
using namespace std;
const int MAXN = 150005;
int a[MAXN],p[MAXN];
// p[i]下一个位置
int n;
int main(){
    memset(p, -1, sizeof(p));
    cin >> n;
```

```
    for (int i = 0; i <= n;i++){
        cin >> a[i];
    }
    // 从后往前推，避免0位置的讨论
    for (int i = n - 3; i >= 0;i--){
        int t = i + 3;
        if(a[i+2]<=a[i+3])
            t = i + 2;
        if(a[i+1]<=a[t])
            t = i + 1;
        a[i] = a[i] + a[t];
        p[i] = t;  // 跳到下一个最近的位置，如果有0的位置，则必会跳
    }
    cout << a[0] << endl;
    int t = 0;
    while(t!=-1){
        if(t)
            cout << " ";
        cout << t;
        t = p[t];
    }
    return 0;
}
```

> 常见题型 from LF：
>
> 第一种是 给你一段长度，在这一段中给出m个点，然后在这m个点中选出k个点，让这k个点之间相邻两个点的之间距离的最大值最小
>
> 第二种是 长为L的桥，上面有M个石子，青蛙从1点开始，每次跳的范围为[S,T]，求过河所要踩的石子数的最小值。（L范围1e9 石头数目不超过100）
>
> 第三种，如这道题倒着过来进行求解。
> 它这个倒着主要是因为 要满足字典序，如果你正着的话，字典序只有把前面存完。因为你只存最末位不一定是最优的。但是你从后往前的话，答案更新相同按照更靠前的更新，假设F[i+1]==F[i+2]，我们这里会选择i+1,它的正确性很明显，因为不管后面i+1和i+2是到哪里，它的首位已经比i+2更优秀了。

7-3 最长公共子序列长度

求两个字符串的最长公共子序列长度。

输入格式:

输入长度≤100的两个字符串。

输出格式:

输出两个字符串的最长公共子序列长度。

输入样例1:

```
ABCBDAB
BDCABA
```

输出样例1:

```
4
```

输入样例2:

```
ABACDEF
PGHIK
```

输出样例2:

```
0
```

代码:

> 比较简单，原理课上讲了的，递推公式直接明朗

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 105;
int c[MAXN][MAXN];
int main(){
    string s1, s2;
    getline(cin, s1);
    getline(cin, s2);
    int m = s1.length(), n = s2.length();
    for (int i = 1; i <= m;i++){
        for (int j = 1; j <= n;j++){
            if(s1[i-1] == s2[j-1])
                c[i][j] = c[i - 1][j - 1] + 1;
            else c[i][j] = max(c[i][j - 1],c[i-1][j]);
        }
    }
    cout << c[m][n];
    return 0;
}
```

https://blog.csdn.net/sm20170867238/article/details/89946300

week16

做对得全分，少选得一半分，多选不得分

3-1 下面有几组货币，每组含有不同面值的货币且每种货币数量没有限制。找零钱问题就是求支付任意的金额（零钱）所需的最少数量货币。选出用贪心法能够获得最优解的货币组。 (20分)

- [x] A. 1, 5,10,20,50,100
- [x] B. 1,10000
- [x] C. 1,3,8,10
- [x] D. 1,3,5,7,11
- [ ] E. 1,4,5,9,14

3-1 答案错误 ⓘ (0 分)

正确答案：ABD

> 暂时想到的比较好的解法：暴力列举第二小~第二大的倍数（但是粗心错过了C）
>
> C：16 = 8 x 2 = 10 x 1 + 3 x 2．故贪心选择出错
>
> E： 8 = 4 x 2 = 5 x 1 + 1 x 3. 故贪心选择出错

【专题】动态规划

Letter-moving Game

> 参考：https://blog.csdn.net/qq_41562704/article/details/102453598

解题思路：其实题目的意思就是求两个字符串的最长公共子序列。我们要将word2恢复到word1的状态，要找到word2与word1最长公共子序列，可以用暴力搜索，最后用word1的长度减去公共子序列的长度就可以了。

```
iononmrogdg
goodmorning
```

```
// 暴力解法
#include<bits/stdc++.h>
using namespace std;
int main(){
    string word1, word2;
    cin >> word1 >> word2;
    int m = 0;
    for(int i = 0; i < word2.length(); ++ i){
        int k = 0, j = i;
```

```
        for(char c : word1)
            if(word2[j] == c){
                ++ j;
                ++ k;
                // printf("%c",c);
            }
        // printf("\n");
        m = max(m, k);
    }
    printf("%d\n",m);
    printf("%d", word1.length()-m);
}
```