

信奥算法

最长公共子序列

Longest Common Subsequence

最长公共子序列

A	G	G	T	A	T	
G	C	T	C	A	C	T

A	G	G	T	A	T	
G	C	T	C	A	C	T

DNA相似度

DNA is built from A(adenine), C(cytosine), G(guanine), and T(thymine). Finding the longest common subsequence between DNA sequences is one of the basic problems in modern computational molecular biology.

输入两条DNA序列，都是由ACGT四个字母组成的字符串。
求它们公共子序列的最长长度。

输入样例
AGGTAT
GCTCACT

输出样例
4

输入样例
AACC
TTCACATT

输出样例
2

输入样例
ACGTACGTACGT
TGCATGCATGCA

输出样例
5

最长公共子序列

LCS 最长公共子序列

输入两条字符串，长度分别为 n 和 m ，
求它们公共子序列的最长长度。

状态
定义

请同学写出 $f[i][j]$ 表示什么含义？

$f[i][j]$ 表示第一条字符串前 i 位
和第二条字符串前 j 位中的LCS

LCS 最长公共子序列

转移
方程

计算 $f[i][j]$ 时依赖哪些格子

若 $a[i] == b[j]$

$$f[i][j] = f[i-1][j-1] + 1$$

若 $a[i] != b[j]$

$$f[i][j] = \max(f[i][j-1], f[i-1][j])$$

```
22 string a,b;
23 cin>>a>>b;
24 int n=a.size();
25 int m=b.size();
26 a=" "+a; b=" "+b; ←
27 for(int i=1;i<=n;i++){
28     for(int j=1;j<=m;j++){
29         if(a[i]==b[j])
30             f[i][j]=f[i-1][j-1]+1;
31         else
32             f[i][j]=
33     }
34 }
35 cout<<f[n][m]<<endl;
```

快快编程

909

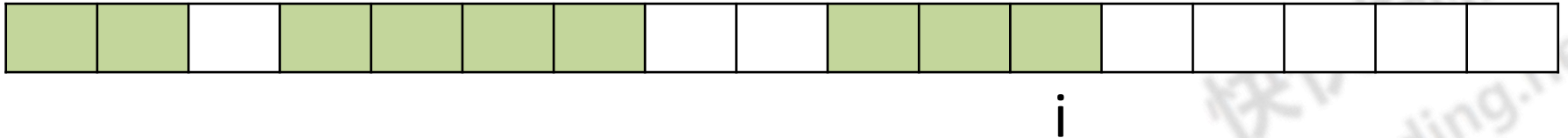
动态规划 - 状态

定义状态

$f[p][i]$ 代表前 i 天 分成不超过 p 段时的最大盈利

$f[p][i]$ 代表以第 i 天结尾 分成恰好 p 段时的最大盈利

$f[p][i]$ 代表以第 i 天结尾 分成不超过 p 段时的最大盈利



动态规划 - 决策/状态转移

$f[p][i]$ 代表以第 i 天结尾 分成不超过 p 段时的最大盈利

决策：第 i 天结尾的那段投资一共连续 len 天

枚举： $len=1,2,3,\dots,i$

再决策：前一段以第 j 天结尾

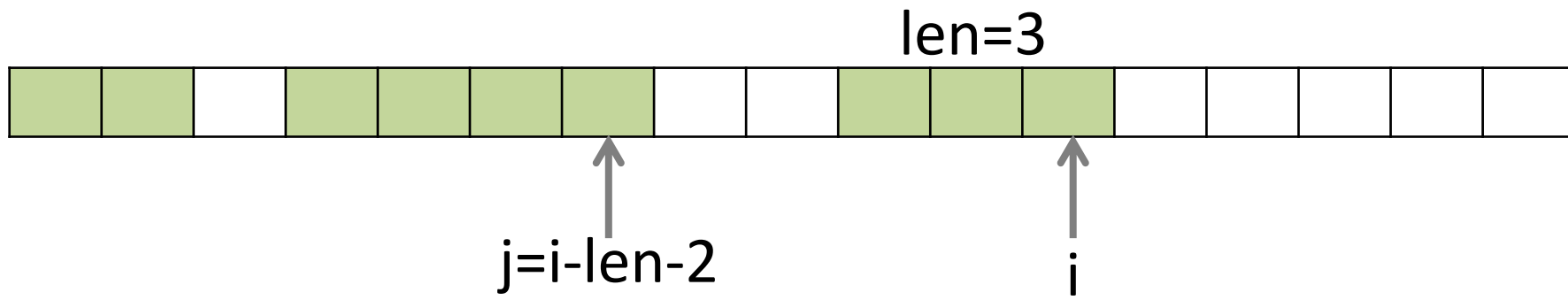
枚举： $j=i-len-1,i-len-2,\dots,1$

复杂度？

$O(N^4)$

状态转移：

$\max\{f[p-1][j]+s[i]-s[i-len]\}$



动态规划 - 决策/状态转移

$f[p][i]$ 代表以第 i 天结尾 分成不超过 p 段时的最大盈利

决策：第 i 天是否单独形成一段投资

第 i 天不是单独1天

状态转移： $x[i] + f[p][i-1]$

第 i 天是单独1天

再决策：前一段以第 j 天结尾
枚举： $j = i-1, i-2, \dots, 1$

状态转移：
 $\max\{x[i] + f[p-1][j]\}$

$O(N^3)$

代码0

```
for(int i=1;i<=n;i++) f[1][i]=x[i]+max(0,f[1][i-1]);
for(int p=2;p<=m;p++)
    for(int i=1;i<=n;i++){
        f[p][i]=x[i]+f[p][i-1];
        for(int j=1;j<=i-1;j++)
            f[p][i]=max(f[p][i],x[i]+f[p-1][j]);
    }
cout<<max(0,*max_element(f[m]+1,f[m]+1+n))<<endl;
```

该代码能得几分?

TE,ME

二维填表

$f[p][i]$ 代表以第 i 天结尾 分成不超过 p 段时的最大盈利

	$x[0]=0$	$x[1]=3$	$x[2]=-1$	$x[3]=1$	$x[4]=-2$	$x[5]=1$
$f[p][i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$p=1$						
$p=2$						
$p=3$						

二维填表

$f[p][i]$ 代表以第 i 天结尾 分成不超过 p 段时的最大盈利

	$x[0]=0$	$x[1]=3$	$x[2]=-1$	$x[3]=1$	$x[4]=-2$	$x[5]=1$
$f[p][i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$p=1$	0	3	2	3	1	2
$p=2$	0	3	2	4	2	4
$p=3$	0	3	2	4	2	???

依赖关系

填表顺序

决策优化

前缀最大值

代码1

```
for(int i=1;i<=n;i++) f[1][i]=x[i]+max(0,f[1][i-1]);
for(int p=2;p<=m;p++){
    int MAX=0;
    for(int i=1;i<=n;i++){
        MAX=max(MAX,f[p-1][i-1]);
        f[p][i]=x[i]+max(MAX,f[p][i-1]);
    }
}
cout<<max(0,*max_element(f[m]+1,f[m]+1+n))<<endl;
```

担心
ME
TE

快快编程
kkcoding.net

二维填表

$f[p][i]$ 代表以第 i 天结尾 分成不超过 p 段时的最大盈利

	$x[0]=0$	$x[1]=3$	$x[2]=-1$	$x[3]=1$	$x[4]=-2$	$x[5]=1$
$f[p][i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$p=1$	0	3	2	3	1	2
$p=2$	0	3	2	4	2	4
$p=3$	0	3	2	4	2	???

依赖关系

填表顺序

滚动数组

代码2

```
for(int i=1;i<=n;i++) f[1][i]=x[i]+max(0,f[1][i-1]);
for(int p=2;p<=m;p++){
    int MAX=0;
    for(int i=1;i<=n;i++){
        MAX=max(MAX,f[p&1^1][i-1]);
        f[p&1][i]=x[i]+max(MAX,f[p&1][i-1]);
    }
}
cout<<max(0,*max_element(f[m&1]+1,f[m&1]+1+n))<<endl;
```

担心
TE

快快编程
kkcoding.net

二维填表

$f[p][i]$ 代表以第 i 天结尾 分成不超过 p 段时的最大盈利

	$x[0]=0$	$x[1]=3$	$x[2]=-1$	$x[3]=1$	$x[4]=-2$	$x[5]=1$
$f[p][i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$p=1$	0	3	2	3		
$p=2$	0	3	2	4	2	
$p=3$	0	3	2	4	2	???

依赖关系


填表顺序

状态优化

无效状态
不用填

代码3

```
for(int i=1;i<=n;i++) f[1][i]=x[i]+max(0,f[1][i-1]);
for(int p=2;p<=m;p++){
    int MAX=0;
    for(int i=1;i<=n-m+p;i++){
        MAX=max(MAX,f[p&1^1][i-1]);
        f[p&1][i]=x[i]+max(MAX,f[p&1][i-1]);
    }
}
cout<<max(0,*max_element(f[m&1]+1,f[m&1]+1+n))<<endl;
```



AC

时间复杂度

$O(nm)$

快快编程
kkcoding.net

动态规划解题顺序

定义状态

枚举决策
状态转移

手动填表

依赖关系

填表顺序

状态优化

决策优化

滚动数组

另外的加速方法

相邻的同号数字可以合并

2 3 -1 -2 5 6 -1 -3 4 4



5 -3 11 -4 8

418怒海

输入3个正整数 n, m, k (≤ 2000 , $m \leq n+2$) 正整数 $a_1..a_n$ (≤ 10000)
输出最多拿的古董总价值。如没可行方案，输出-1

输入样例

5 3 2

1 2 3 4 10

输出样例

6

输入样例

10 5 3

1 2 3 4 5 6 7 8 9 10

输出样例

28

无法到达海面?

若 $n+1 > m * k$ 则无法到达海面

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

小心越界

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

输入样例

5 3 2

1 2 3 4 10

每一格
都依赖
左上方

填表顺
序如何
选择？

a[i]		j=0	j=1	j=2	j=3
0	i=0	0	0	0	0
1	i=1	0	1	0	0
2	i=2	0	2	3	0
3	i=3	0	0	5	6
4	i=4	0	0	6	9
10	i=5	0	0	0	16
0	i=6	0	0	0	6

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

```
9   for(int i=1;i<=n+1;i++)
10       for(int j=(i+k-1)/k;j<=m&& j<=i;j++)
11           for(int p=i-1;p>=0&&p>=i-k;p--)
12               f[i][j]=max(f[i][j],f[p][j-1]+a[i]);
```

枚举行数i从小到大

枚举列数j从小到大

枚举决策p上浮前在哪一层

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

```
9  for(int j=1;j<=m;j++)
10     for(int i=1;i<=n+1&& i<=j*k;i++)
11         for(int p=i-1;p>=0&&p>=i-k;p--)
12             f[i][j]=max(f[i][j],f[p][j-1]+a[i]);
```

枚举列数j从小到大

枚举行数i从小到大

枚举决策p上浮前在哪一层

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

输入样例

5 3 2

1 2 3 4 10

每一格
只依赖
左侧k格

滑动窗
口最值

a[i]		j=0	j=1	j=2	j=3
0	i=0	0	0	0	0
1	i=1	0	1	0	0
2	i=2	0	2	3	0
3	i=3	0	0	5	6
4	i=4	0	0	6	9
10	i=5	0	0	0	16
0	i=6	0	0	0	6

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

输入样例

5 3 2

1 2 3 4 10

每一格
只依赖
左侧k格

滑动窗
口最值

a[i]		j=0	j=1	j=2	j=3
0	i=0	0	0	0	0
1	i=1	0	1	0	0
2	i=2	0	2	3	0
3	i=3	0	0	5	6
4	i=4	0	0	6	9
10	i=5	0	0	0	16
0	i=6	0	0	0	6

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

输入样例

5 3 2

1 2 3 4 10

每一格
只依赖
左侧k格

滑动窗
口最值

a[i]		j=0	j=1	j=2	j=3
0	i=0	0	0	0	0
1	i=1	0	1	0	0
2	i=2	0	2	3	0
3	i=3	0	0	5	6
4	i=4	0	0	6	9
10	i=5	0	0	0	16
0	i=6	0	0	0	6

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

输入样例

5 3 2

1 2 3 4 10

每一格
只依赖
左侧k格

滑动窗
口最值

a[i]		j=0	j=1	j=2	j=3
0	i=0	0	0	0	0
1	i=1	0	1	0	0
2	i=2	0	2	3	0
3	i=3	0	0	5	6
4	i=4	0	0	6	9
10	i=5	0	0	0	16
0	i=6	0	0	0	6

$f[i][j]$ 代表到 i 层已用 j 次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

```
9
10
11
12
13
14
15
16
17
18
19

for(int j=1;j<=m;j++){
    int l=0,r=0;
    for(int i=0;i<=n+1&& i<=j*k;i++){
        while(l<r&&i-q[l]>=k)l++;
        while(l<r&&f[i][j-1]>f[q[r-1]][j-1])r--;
        q[r++]=i;
        MAX[i]=f[q[l]][j-1];
    }
    for(int i=1;i<=n+1&&i<=j*k;i++)
        f[i][j]=MAX[i-1]+a[i];
}
```

单调队列

滑动窗口最值

$f[i][j]$ 代表到i层已用j次上浮时最多拿到多少价值的古董

$$f[i][j] = \max_{i-k \leq p \leq i-1} \{f[p][j-1]\} + a[i]$$

输入样例

5 5 4

1 2 3 4 5

寻找无
效状态

a[i]		j=0	j=1	j=2	j=3	j=4	j=5
0	i=0	0	0	0	0	0	0
1	i=1	0	1	0	0	0	0
2	i=2	0	2	3	0	0	0
3	i=3	0	3	5	6	0	0
4	i=4	0	4	7	9	10	0
5	i=5	0	5	9	12	14	15
0	i=6	0	0	4	9	12	14

无义
状态

无关
状态

剔除无效状态

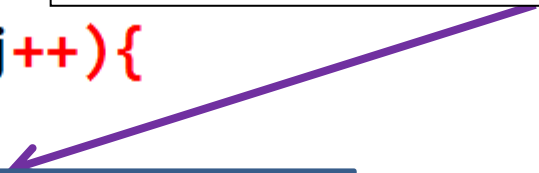
`i=j; i<=n+1+j-m`

```
9 | for(int j=1; j<=m; j++)  
10 |     for(int i=j; i<=n+1+j-m && i<=j*k; i++)  
11 |         for(int p=i-1; p>=0 && p>=i-k; p--)  
12 |             f[i][j]=max(f[i][j], f[p][j-1]+a[i]);
```

剔除无效状态

$i = j - 1; i \leq n + 1 + j - m$

```
9  for(int j=1;j<=m;j++){
10     int l=0,r=0;
11     for(int i=j-1;i<=n+1+j-m&& i<=j*k;i++){
12         while(l<r&&i-q[l]>=k)l++;
13         while(l<r&&f[i][j-1]>f[q[r-1]][j-1])r--;
14         q[r++]=i;
15         MAX[i]=f[q[l]][j-1];
16     }
17     for(int i=j;i<=n+1+j-m&&i<=j*k;i++)
18         f[i][j]=MAX[i-1]+a[i];
19 }
```



快快编程作业

909

418

191

拓展题

916