线性DP

线性DP指的是具有线性阶段划分的动态规划。

最长上升子序列

最长公共子序列

数字三角形

P1868 饥饿的奶牛

线性DP板子题。

用 f_i 表示前 i 个位置能吃到最多的牧草，若有区间 (i,j)，则易得到转移：

$$f_j=\max_i(f_{i-1}+(j-i+1))$$

用 vector 预处理出 j 对应的 i 即可。

P4310 绝世好题

用 f_i 表示仅考虑前 i 位的答案，易得到 DP 方程：

$$f_i=\max_{j<i\ b_i\&b_j\neq 0} f_j+1$$

显然是 $O(n^2)$ 的复杂度，考虑简化状态以减少决策集合的大小来降低时间复杂度。

发现每次转移时，b_i 和 b_j 仅需要有一个二进制位相同即可转移，所以用 $f_{i,j}$ 表示前 i 位选择的最后一个数第 j 位为1 时的答案，则有：

$$f_{i,k}=\max_j f_{i-1,j}+1\ (b_i\text{ 的第 }j,k\text{ 位为 }1)$$

容易发现 i 这一维是可以去掉的。

在动态规划中，减少决策集合的大小是一种常见的策略，比如：CF10D LCIS （注意此题线性 DP 可以做到 $O(n^2)$）

P4158 [SCOI2009]粉刷匠

需要 DP 两次的题。

令 $f_{i,j}$ 表示前 i 块版刷 j 次能正确刷的最多格子，$g_{i,j,k}$ 表示第 i 块板前 k 格粉刷 j 次能最多粉刷的格子数。

对于 f ，显然有：

$$f_{i,j} = \max_k(f_{i-1,j-k} + g_{i,k,m})$$

然后考虑求 g，枚举一个 q，比较 $[q+1,k]$ 刷哪种颜色更划算再转移。

$$g_{i,j,k} = \max_q(g_{i,j-1,q} + \max(sum_{i,k} - sum_{i,q}, k - q - (sum_{i,k} - sum_{i,q})))$$

（$sum_{i,j}$ 表示 i 行前 j 格的蓝色/红色个数）

P3558 [POI2013]BAJ-Bytecomputer

暴力分类讨论题。

用 $f_{i,j}$ 表示前 i 个已经单调不降时，第 i 个数为 $j-1$ 需要的最少操作数。

当 $a_i = -1$ 时：

如果 a_i 将会改为 -1，所以不需要操作，并且第 $i-1$ 位只能是 -1：

$$f_{i,0} = f_{i-1,0}$$

如果 a_i 将会改为 0，由于单调不降性，第 $i-1$ 位必须初始为 1 才能转移，显然此时需要一次操作：

$$f_{i,1} = \min(f_{i-1,0}, f_{i-1,1}) + 1(a_{i-1} = 1)$$

$$f_{i,1} = inf(a_{i-1} \neq 1)$$

如果 a_i 将会改为 1，如果 $a_{i-1} = 1$，则可以在任意时刻转移，反之只能在前一位修改为 1 时才能修改，显然需要两次操作：

$$f_{i,2} = \min(f_{i-1,0}, f_{i-1,1}, f_{i-1,2} + 2)(a_{i-1} = 1)$$

$$f_{i,2} = f_{i-1,2} + 2$$

同理可以求出 $a_i = 1$ 时和 $a_i = 2$ 时的方程。

推荐题目：P3336 [ZJOI2013]话旧（更有思维难度的分类讨论，基本的组合知识，细节处理）

CCPC2023 秦皇岛F Mystery of Prime(Mystery of Prime - Problem - QOJ.ac)（先发现性质才能找到 DP 状态）

蒟蒻的题解：Link（话旧）

[P2501 [HAOI2006]数字序列](#)

先考虑第一问。
显然，要求 $a_i - a_{i-1} >= 1$，即 $a_i - a_j >= i - j$，移项得 $a_i - i >= a_j - j$ （$i > j$）。
所以构造一个数列 b 使 $b_i = a_i - i$，那么 b 的最长不降子序列以外的数要改。
然后考虑第二问。
对于其中已经单调不降的最长子序列，令子序列中两个相邻的项对应原序列的下标为 i，j （$i < j$），那么原序列 i，j 之间的点就需要修改。
对于任意一种合法的方案，我们把一些最长连续并且值相同的项称为台阶（台阶的长度可以为1）。
对于每个台阶，有两个量，上升值表示台阶中值增加的项的个数，下降值表示台阶中值减少的项的个数。
如果下降值小于上升值，则将台阶中所有项的值变为左边台阶的值就可以是变化的值减少。
反之，将台阶中所有的值变为右边台阶的值。
所以对于 i，j 之间的数修改的最优方案（之一），一定有一个 k （$i \leq k < j$），使得 i 到 k 内的值相等，$k + 1$ 到 j 的值相等。
对于每个 i，j 枚举 k 即可，用 DP 统计答案。

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int Maxn=1e5+5;
inline int read(){
    int s=0,w=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
    while(ch>='0'&&ch<='9')s=(s<<1)+(s<<3)+ch-'0',ch=getchar();
    return s*w;
}
int n,b[Maxn],f[Maxn],len,c[Maxn],dp[Maxn],suml[Maxn],sumr[Maxn];
vector <int> vec[Maxn];
signed main(){
    n=read();
    for(int i=1;i<=n;i++)b[i]=read()-i;
    b[n+1]=1e9;
    for(int i=1;i<=n+1;i++){
        int l=0,r=len;
        while(l<r){
            int mid=(l+r+1)>>1;
            if(f[mid]<=b[i])l=mid;
            else r=mid-1;
        }
        if(l==len)++len;
        f[l+1]=b[i];
        c[i]=l+1;
        vec[c[i]].push_back(i);
    }
    vec[0].push_back(0);
    b[0]=-1e9;
    memset(f,20,sizeof(f));f[0]=0;
    for(int i=1;i<=n+1;i++){
        for(int j=0;j<vec[c[i]-1].size();j++){
            int x=vec[c[i]-1][j];
            if(x>i||b[x]>b[i])continue;
            suml[x]=sumr[i-1]=0;
            for(int k=x+1;k<=i-1;k++)suml[k]=suml[k-1]+abs(b[x]-b[k]);
            for(int k=i-2;k>=x;k--)sumr[k]=sumr[k+1]+abs(b[i]-b[k+1]);
            for(int k=x;k<=i-1;k++)f[i]=min(f[i],f[x]+suml[k]+sumr[k]);
```

```
        }
    }
    printf("%lld\n%lld",n-len+1,f[n+1]);
    return 0;
}
```