L2C1 Lesson2

> DP状态计算划分依据扩展：最后一个不同的点位
>
> 最长上升子序列模型（LIS问题）
>
> - 怪盗基德的滑翔翼 （初始时选择任一位置任意方向，中途不能改改变方向）
>   - 确定起点和方向之后，退化为LIS问题
>
>   起点：a[i]
>
>   最长距离：：以a[i]结尾的最长上升子序列（俩个方向上分别求解LIS问题）

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 110;

int n;
int a[N], f[N];

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        scanf("%d", &n);
        for (int i=1; i<=n; i++) scanf("%d", &a[i]);

        int res = 0;
        //正向求解LIS
        for (int i=1; i<=n; i++) {  //以a[i]结尾的LIS解
            f[i] = 1;   //前一个数字为空
            for (int j=1; j<i; j++)
                if (a[i] > a[j])
                    f[i] = max(f[i], f[j]+1);
            res = max(res, f[i]);
        }

        //反向求解LIS
        for (int i=n; i>=1; i--) {
            f[i] = 1;
            for (int j=n; j>i; j--)
                if (a[i] > a[j])
                    f[i] = max(f[i], f[j]+1);
            res = max(res, f[i]);
        }

        printf("%d\n", res);
    }

    return 0;
```

```
41  }
```

- 登山问题
  - 条件1：按照编号递增顺序浏览 => 子序列
  - 条件2：相邻两个景点海拔不能相同
  - 条件3：一旦开始下降，就不能上升 => 走过的路线先严格单调上升后严格单调下降
  - 目标：求最多能浏览多少景点
    - 所有形状满足上述条件的子序列长度的最大值
  - 思路：
    - 状态表示
      - 集合：所有顶点是a[i]且满足上述条件的子序列集合
      - 属性：ff[i]子序列集合的长度最大值
    - 状态计算
      - 性质：对于ff[i]所代表的子序列集合而言，a[k]左右两边互相独立，故左右两边分别求最大值
        - 左边：所有以a[k]结尾的LIS问题的长度的最大值
        - 右边：所有反向以a[k]结尾的LIS问题最大值
      - 所以对每个点a[i]的正反向LIS答案进行预处理（优化时间复杂度）求解

        f[i]表示a[i]结尾正向LIS答案，g[i]表示a[i]结尾反向LIS答案。则ff[i] = f[i]+g[i]-1

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 1010;

int n;
int a[N];
int f[N], g[N];

int main() {
    scanf("%d", &n);

    for (int i=1; i<=n; i++) scanf("%d", &a[i]);

    //预处理正向LIS
    for (int i=1; i<=n; i++) {
        f[i] = 1;
        for (int j=1; j<i; j++)
            if (a[i] > a[j])
                f[i] = max(f[i], f[j]+1);
    }

    //预处理反向LIS
    for (int i=n; i>=1; i--) {
```

```
            g[i] = 1;
            for (int j=n; j>i; j--)
                if (a[i] > a[j])
                    g[i] = max(g[i], g[j]+1);
        }

        int res = 0;
        for (int k=1; k<=n; k++) res = max(res, f[k]+g[k]-1);

        printf("%d\n", res);

        return 0;
    }
```

- 合唱队行（登山问题变形，队友问题）
  - 去掉的人数最少，即合唱队形最长，可以先求出最长合唱队形，再由总数减去该值即为答案

```
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 110;

int n;
int a[N];
int f[N], g[N];

int main() {
    scanf("%d", &n);
    for (int i=1; i<=n; i++) scanf("%d", &a[i]);

    for (int i=1; i<=n; i++) {
        f[i] = 1;
        for (int j=1; j<i; j++)
            if (a[i] > a[j])
                f[i] = max(f[i], f[j]+1);
    }

    for (int i=n; i>=1; i--) {
        g[i] = 1;
        for (int j=n; j>i; j--)
            if (a[i] > a[j])
                g[i] = max(g[i], g[j]+1);
    }

    int res = 0;
    for (int k=1; k<=n; k++) res = max(res, f[k]+g[k]-1);

    printf("%d", n-res);

```

```
    return 0;
}

```

- 友好城市
  - 条件1：每个城市上只能建立一座桥
  - 条件2：只能在友好城市之间建桥
  - 条件3：所有桥与桥之间不能相交
  - 目标：最多可以建多少桥
  - 思路：
    - 将两岸分开考虑，以南侧为自变量考虑，将（南侧城市坐标，对应北侧友好城市坐标）按照第一个关键字从小到大排序，可发现所有合法建桥方案的北侧城市的坐标序列必须是上升子序列，且因变量北侧城市的每一个上升子序列都对应一个合法的建桥方式，因此上升子序列长度的最大值即为答案。

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

typedef pair<int, int> PII;

const int N = 5010;

int n;
PII q[N];
int f[N];

int main() {
    scanf("%d", &n);
    for (int i=0; i<n; i++) scanf("%d%d", &q[i].first, &q[i].second);

    sort(q, q+n);   //按南岸城市坐标从小到大排序

    int res = 0;
    for (int i=0; i<n; i++) {
        f[i] = 1;
        for (int j=0; j<i; j++)
            if (q[i].second > q[j].second)
                f[i] = max(f[i], f[j]+1);

        res = max(res, f[i]);
    }

    printf("%d", res);

    return 0;
}
```

```
34
```

- 最大上升子序列和
  - 状态表示：f[i]
    - 集合：所有以a[i]结尾的上升子序列
    - 属性：子序列和的最大值
  - 状态计算：按子序列倒数第二个数进行分类，假设倒数第二个数是a[k]，则所有倒数第二个数是a[k]的子序列的最大值是**f[k]+a[i]**，其他类类似，因此，f[i] = max(f[j]+a[i])，且满足a[j] < a[i]，j=0, ..., i-1

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 1010;

int n;
int a[N];
int f[N];

int main() {
    scanf("%d", &n);

    for (int i=1; i<=n; i++) scanf("%d", &a[i]);

    int res = 0;
    for (int i=1; i<=n; i++) {
        f[i] = a[i];
        for (int j=1; j<i; j++)
            if (a[i] > a[j])
                f[i] = max(f[i], f[j]+a[i]);    //注意是a[i]

        res = max(res, f[i]);
    }

    printf("%d", res);

    return 0;
}
```

- 拦截导弹（LIS+贪心）
- 导弹防御系统
- 最长公共上升子序列