Minimum number of jumps

> 给定数组arr，arr[i]表示在当前位置i，所能够跳到的下一个位置，求从i=0开始跳到数组末尾需要的最小跳步骤数

贪心

curMaxPosition表示在当前位置i，所能跳跃达到的最远位置；preMaxPosition表示之前所能够跳跃达到的最远位置

这里贪的是一个能到达的最远范围，遍历当前跳跃能到的所有位置，然后根据该位置上的跳力来预测下一步能跳到的最远距离，贪出一个最远的范围，一旦当这个范围到达末尾时，当前所用的步数一定是最小步数。

只要curMaxPosition未达到最后一个位置则循环继续，preMaxPosition先赋值为curMaxPosition的值，表示上一次跳跃后能到达的最远位置

如果当前位置i小于等于preMaxPosition，说明还是在上一跳能到达的范围内，根据当前位置加跳力来更新curMaxPosition

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

using namespace std;

using vi = vector<int>;

int minJumpsToEnd(vi &arr, int N) {
    if (arr.empty()) return 0;
    int minJumps = 0, i = 0, curMaxPosition = 0;
    while (curMaxPosition < N - 1) {
        minJumps++;
        int preMaxPosition = curMaxPosition;
        while (i <= preMaxPosition) {
            curMaxPosition = max(curMaxPosition, i + arr[i]);
            i++;
        }
    }
    return minJumps;
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N;
        scanf("%d", &N);
        vi arr(N);
        for (int i = 0; i < N; ++i) scanf("%d", &arr[i]);
        printf("%d\n", minJumpsToEnd(arr, N));
    }
    return 0;
}
```

动态规划

jumps[i]表示达到位置i所需要的最小步骤数

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

using namespace std;

int minJumpsToEnd(vector<int> &arr, int N) {
    if (arr[0] == 0 || N == 0) return -1;
    vector<int> jumps(N, INT_MAX);
    jumps[0] = 0;
    for (int i = 1; i < N; ++i) {
        for (int j = 0; j < i; ++j) {
            if (i - j <= arr[j] && jumps[j] != INT_MAX) {
                jumps[i] = min(jumps[i], jumps[j] + 1);
                break;
            }
        }
    }
    if (jumps[N - 1] == INT_MAX) return -1;
    return jumps[N - 1] ;
}
```

```
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N;
        scanf("%d", &N);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i) scanf("%d", &arr[i]);
        printf("%d\n", minJumpsToEnd(arr, N));
    }
    return 0;
}
```