

## 尺取法

通常是具有单调特征序列、寻找连续子区间接近目标值的问题中应用。关注左右两端的指针，当选定的区间中所关注的值大于目标值时将左侧指针左移，当选定的区间中所关注的值小于目标值时将右指针右移，直至终点。在这个过程中枚举的区间的子区间的集合即是序列本来的所有子区间的集合，确保了间接枚举了每个区间，但与直接枚举并不同，一般在只关注符合特征的最短的区间的问题中这样做是正确的。

看几个例题

### poj 3061

给定一个正数序列，使得其和大于或等于 $S$ 求最短的子序列长度。

如果枚举两端的话复杂度就是 $O(n^2)$ 但是正数序列是单调的，也就是说扩大区间其和一定增大，减小区间其和一定减小；并且我们其实不关心每一个区间，我们关心的是和大于 $S$ 的区间，而这样的区间由于单调性肯定是和越短和越小，因此对于同一个左/右位置我们关心的是使和尽可能小的但仍大于 $S$ 的另一端。可能说得不是很明确，不过这道题从直觉上、一定的理性上，尺取法都是一个好的选择，而且将复杂度将至 $O(n)$

思路是，当选取的区间和大于 $S$ 时就把左端点右移直至小于 $S$ 当区间和小于 $S$ 时就将右端点右移直至大于 $S$ 等于的情况合理即可。

```
#include<cstdio>
using namespace std;
int n,s,t,a[100005];
int min(int x,int y)
{
    if(x==-1)
        return x;
    if(x>y)
        return y;
    return x;
}
int main()
{
    scanf("%d",&t);
    while(t-->0)
    {
        scanf("%d%d",&n,&s);
        for(int i=1;i<=n;i++)
            scanf("%d",&a[i]);
        a[n+1]=0;
        int left=1,right=1,length=100005;
        long long sum=a[1];
        while(right<=n)
        {
            if(sum>=s)
            {
                length=min(length,right-left+1);
                sum-=a[left++];
            }
            else
                sum+=a[right++];
        }
        printf("%d\n",length);
    }
}
```

```
        sum-=a[left++];
    }
    else sum+=a[++right];
}
if(length==100005)
    length=0;
printf("%d\n",length);
}
return 0;
}
```

## poj 3320

一本书有  $P$  页，每页都有  $a[i]$  个知识点，知识点可能重复，求最少的连续页数来覆盖所有知识点。

这个题虽然和上面的不同但是知识点种类仍然是单调的，随着区间的增缩而增缩，而且也是要连续页、最少页数，因此用尺取法可以解决。

代码略。

## poj 2566

给定一个数组和一个值  $t$  求一个子区间使得其和的绝对值与  $t$  的差值最小。数组中的数可正可负。

这个题看起来并不符合我们使用尺取法的条件。不过我们可以将其变形。

将前缀和排序，排序后每两个位置的前缀和之差代表一个原来的区间的和的绝对值(妙啊)，同时在这个排序后的前缀和序列中随区间增缩所取的区间和的绝对值也增缩。这样运用尺取法是一个好的选择。

```
#include<cstdio>
#include<algorithm>
#define INF 0x3f3f3f3f
using namespace std;
int n,k,a;
typedef pair<int,int> unit;
unit sum[100005];
int main()
{
    freopen("input.in","r",stdin);
    while(scanf("%d%d",&n,&k)&&n)
    {
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&a);
            sum[i]=unit(sum[i-1].first+a,i);
        }
    }
}
```

```
    sort(sum+1,sum+1+n);
    for(int i=1;i<=k;i++)
    {
        scanf("%d",&a);
        int t=INF,l,r;
        for(int left=1,right=1;right<=n;right++)
        {
            while(sum[right].first-sum[left].first>a)
            {
                if(sum[right].first-sum[left].first-a<t)
                {
                    t=sum[right].first-sum[left].first-a;
                    l=sum[left].second;
                    r=sum[right].second;
                }
                left++;
            }
            if(a-sum[right].first+sum[left].first<t)
            {
                t=a-sum[right].first+sum[left].first;
                l=sum[left].second;
                r=sum[right].second;
            }
        }
        printf("%d %d %d\n",sum[r].first-sum[l].first,l,r);
    }
}
return 0;
}
```

## 总结

运用尺取法一定要注意题目中序列的特征，因为尺取法并不能真正枚举每一个区间。

对序列进行变换也可能让尺取法可以应用。

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:no\\_morning\\_training:%E5%B0%BA%E5%8F%96%E6%B3%95&rev=1588940543](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:no_morning_training:%E5%B0%BA%E5%8F%96%E6%B3%95&rev=1588940543)

Last update: 2020/05/08 20:22