DP with DS

Pick some elements from an array. The minimum distance between two chosen elements should be at least k. Maximize the sum of chosen elements.

Pick some elements from an array. The minimum distance between two chosen elements should be at least k. Maximize the sum of chosen elements.

dp[i] = Maximum answer if ith element is the last chosen element.

Pick some elements from an array. The minimum distance between two chosen elements should be at least k. Maximize the sum of chosen elements.

dp[i] = Maximum answer if ith element is the last chosen element.

```
for(int i=1;i<=n;i++)
{
    dp[i]=a[i];
    if(i-k-1>=1)dp[i]=max(dp[i],a[i]+max_range_query(1,i-k-1));
    update(i,dp[i]);
}

int ans=-inf;
for(int i=1;i<=n;i++)ans=max(ans,dp[i]);

cout<<ans<<endl;
```

Longest Increasing Subsequence

N^2 DP Solution of LIS

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

Longest Increasing Subsequence

N^2 DP Solution of LIS

```
for(int i=1;i<=n;i++)
{
    int x=a[i];
    dp[x]=max(dp[x],1);
    for(int j=1;j<i;j++)
    {
        int y=a[j];
        if(y<x)
        {
            dp[x]=max(dp[x],dp[y]+1);
        }
    }
}
```

Longest Increasing Subsequence

N^2 DP Solution of LIS

{ 10, 22, 9, 33, 21, 50, 41, 60 }

```
for(int i=1;i<=n;i++)
{
    int x=a[i];
    dp[x]=max(dp[x],1);
    for(int j=1;j<i;j++)
    {
        int y=a[j];
        if(y<x)
        {
            dp[x]=max(dp[x],dp[y]+1);
        }
    }
}
```

Longest Increasing Subsequence

NlogN DP Solution of LIS

{ 3, 2 , 5 , 1 , 7 , 3 }

```
for(int i=1;i<=n;i++)
{
    int x=a[i];
    dp[x]=max(dp[x],1);
    dp[x]=max(dp[x],1+range_max_query(1,x-1));
    update(x,dp[x]);
}
```