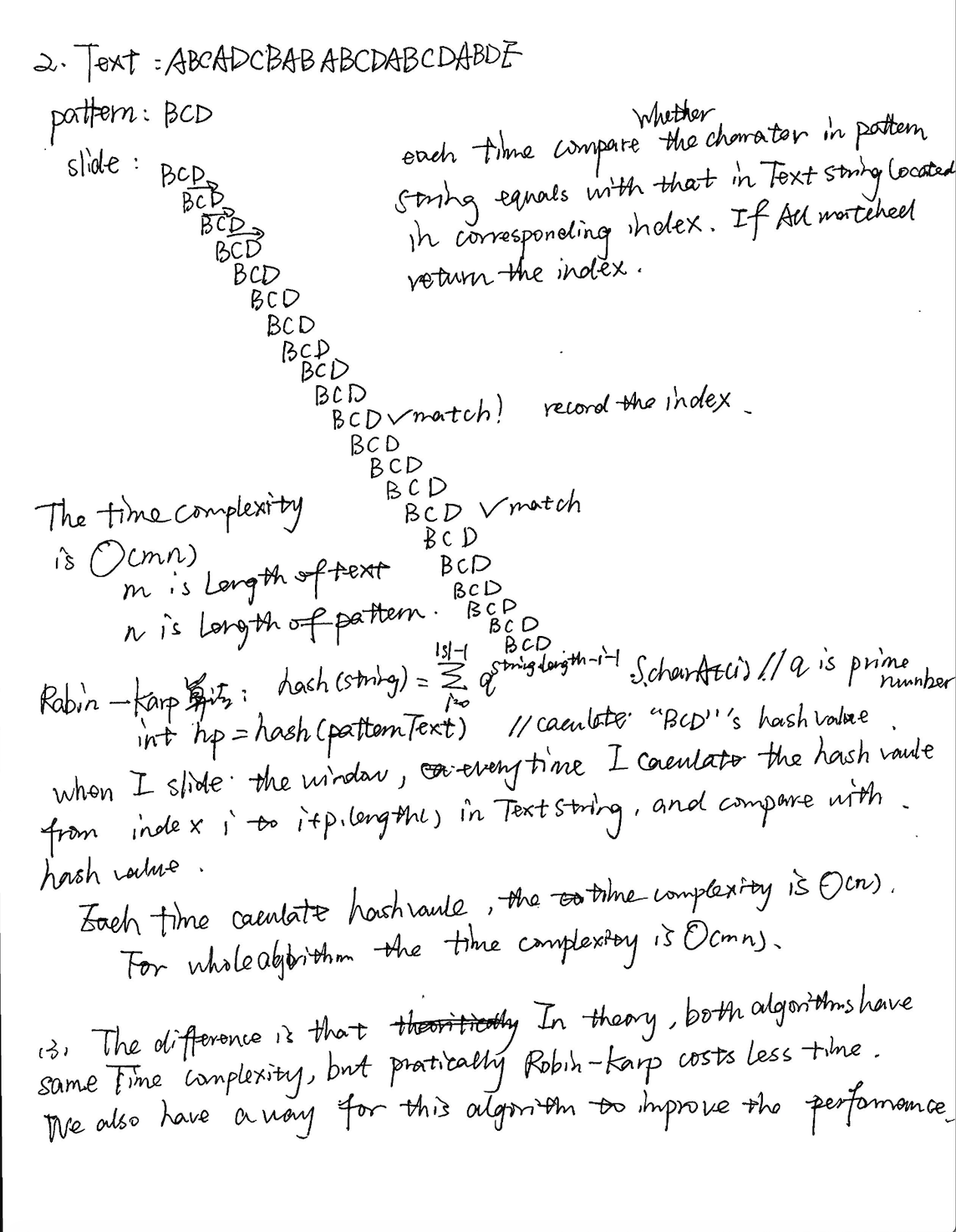
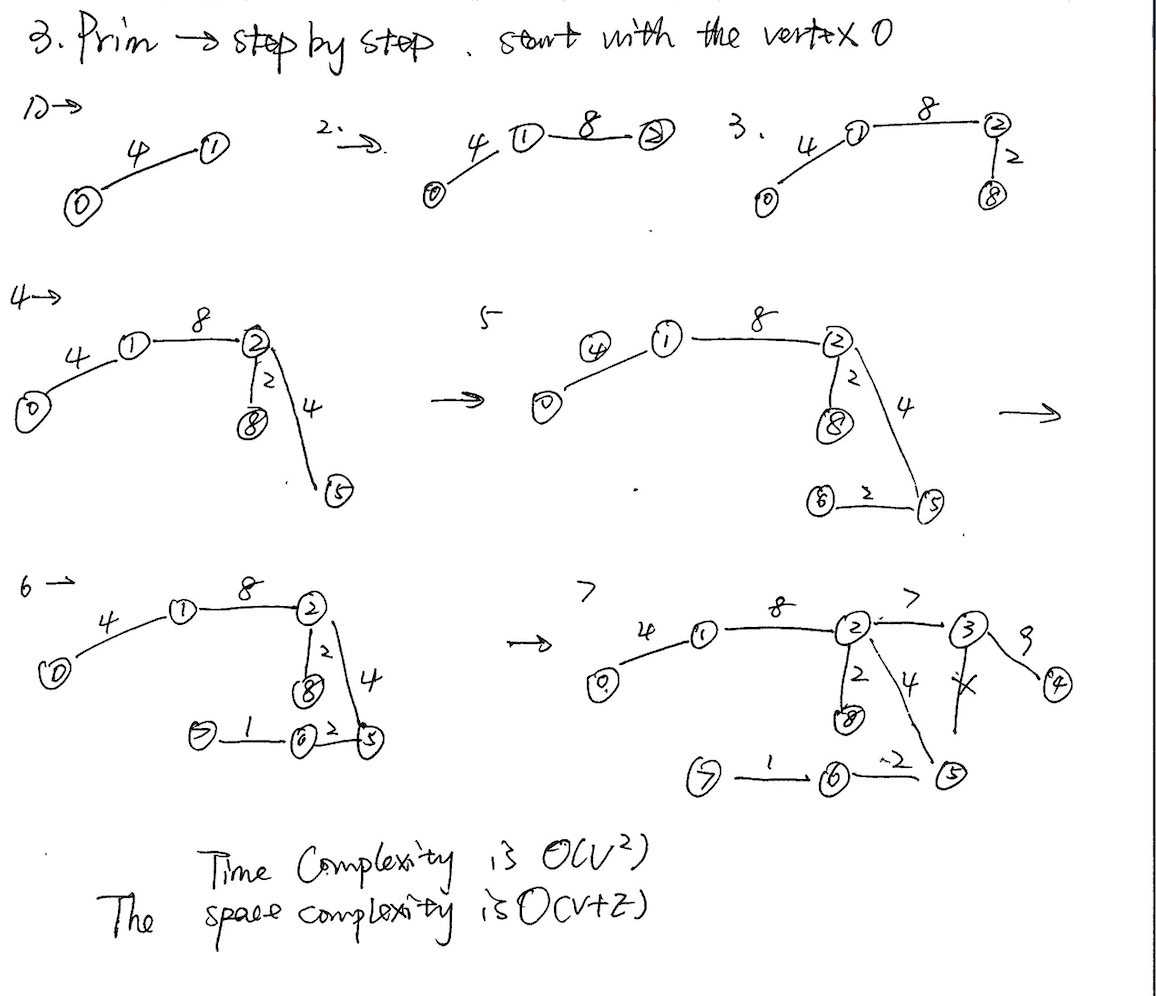
Homework10

2.





4.

The result is 30

: is A method of using dynamic programming to find the optimal value. Suppose dp[N][V] is used to store the intermediate state value, and dp[i][j] represents the maximum value of the sum of the values ​​of the items in the backpack with the capacity of j for the previous i item (note the maximum value). Then we only need to know the value of dp[i=N][j=V], which is the title.

Now consider the state transition equation for dynamically planning the array dp[i][j]:

Suppose we have determined that the maximum value of the value of the backpack of the first i-1 item loading capacity j is dp[i-1][j], and the value of the fixed capacity j does not change, then the method of loading the i-th item Discussed as follows:

First, the weight weight [i] of the i-th item must be less than or equal to the capacity j, that is,

1. If weight[i]>j, the i-th item must not be loaded with a backpack of capacity j, at this time dp[i][j]=dp[i-1][j]

2. If weight[i]<=j, the first thing to be clear is that this item can be loaded into a backpack of capacity j, then if we load the item, then

Dp[i][j]=dp[i-1][j-weight[i]]+value[i]

The ensuing problem is that we have to judge whether the total value in the backpack is the largest after the i-th item is loaded into the backpack with the capacity j. In fact, it is a good judgment, that is, if the total value of the i-th item is dp[i-1][j-weight[i]]+value[i]>the total value before the installation is dp[i-1 ][j], then Ken is the biggest; otherwise it means that the i-th item does not have to be loaded with a backpack of capacity j (the total value becomes smaller after loading, then it is definitely not necessary to install it)

Therefore, the state transition equation is as follows:

Dp[i][j] = (dp[i-1][j] > (dp[i-1][j-weight[i]]+value[i]))? dp[i-1][j ]:(dp[i-1][j-weight[i]]+value[i])