# 动态规划-永远的神！好的动态规划速度很快。

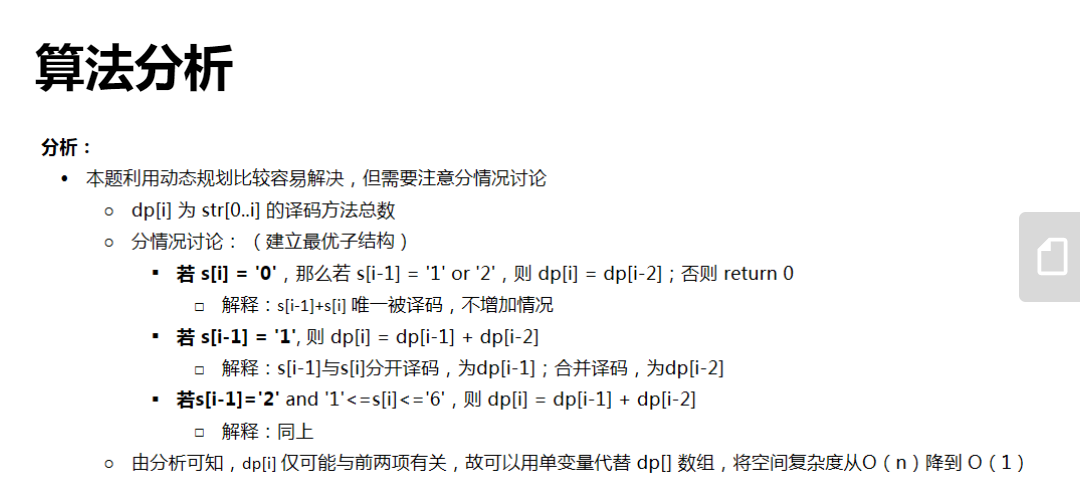
## 题目描述：



## 回溯算法超时：

```  
//注意，"06" 不能映射为 "F" ，因为 "6" 和 "06" 不同。  
//要 解码 已编码的消息，所有数字必须基于上述映射的方法，反向映射回字母（可能有多种方法）。  
  
//用例提示：  
//1：字符串以0开头无映射  
//2：题目数据保证答案肯定是一个 32 位 的整数。  
  
class Solution {  
 private:  
 int res;  
 //回溯算法  
 bool change\_type(string&s,int startIndex,int len){  
 if(len==2&&s[startIndex]=='0')return false;  
 int ans=0;  
 for(int i=startIndex;i<startIndex+len;i++){  
 ans\*=10;  
 ans+=s[i]-'0';  
 }  
 if(ans>26||ans==0)return false;  
 return true;  
 }  
  
 void backtracking(string&s,int startIndex){  
 //终止条件  
 if(startIndex==s.size()){  
 ++res;  
 return ;  
 }  
  
 for(int len=1;len<=2;len++){  
 if(startIndex+len>s.size())break;  
 if(change\_type(s,startIndex,len)){  
 backtracking(s,startIndex+len);  
 }else break;  
 }  
  
 }  
public:  
 int numDecodings(string s) {  
 res=0;  
 backtracking(s,0);  
 return res;  
 }  
};  
```

## 动态规划：尽量降低动态数组的维度



//回溯超时，转向动态规划  
class Solution {  
public:  
 int numDecodings(string s) {  
 int cur=1,pre=1;  
 if(s[0]=='0')return 0;  
 for(int i=1;i<s.size();i++){  
 int temp=cur;  
 if(s[i]=='0'){  
 if(s[i-1]=='1'||s[i-1]=='2')  
 cur=pre;  
 else return 0;  
 }  
 else if(s[i-1]=='1')cur=cur+pre;  
 else if(s[i-1]=='2'&&s[i]<='6'&&s[i]>='1')cur=cur+pre;  
   
 pre=temp;  
 }  
  
 return cur;  
 }  
};