DP

插头dp:

Bzoj2331首先把小的一边放下来，然后考虑插头dp: 设0为无插头，1为没转过弯的，2为转过弯的，然后慢慢讨论即可。

Bzoj1814 即标准的插头dp,设有无插头即可。

好像没有了。

主要解决的问题就是棋盘上的覆盖问题，逐格递推，核心思想：讨论上插头和左插头不同的状态，并不一定要设为左括号右括号，同时注意答案，做的时候细心一点即可，同时为了节约时间和空间可以逐层hash来递推。同时当是路径时加一维无对应匹配插头的状态。

状压dp:

1、题目应符合“具有最优性的重叠子问题”，比如上面的问题都符合：要计算 A 状态的最值（最优性）， 那么可能会多次用到子状态 B 的最优值（重叠性子问题），则把 B 状态的最优值用记忆表记忆，以 后直接引用即可，无须重复计算，这也是动态规划相对与搜索区别：空间换时间。分步（阶段）决策、状态、记忆表、状态转移方程都是动态规划的关键词。

2、问题的状态可以用集合表示，但集合又不能太大，一般是 20 个元素左右！

3、用十进制整数表示集合（十进制对应的二进制），需要把题目的要素按 0,1,…,n-1 编号，用位运算实现集合的运算。

4、状态是集合的动态规划的实质就是：记忆表是记录的某些集合下的方案数或最优值。

一些题目：

Vijosp1188:TSP 问题，哈密顿回路问题

题解：设f[i][state]为当前在i,还有state的状态没有走过的所需的最小步数即可实现记忆化，从0出发，当走完后回到0即可。

Bzoj3195:普通的状压，就是多设一维定序即可。

Bzoj2669:核心思想：状压局部最小值的位置即可。

还有些状压dp的题，但总结起来没什么意思，解法千奇百怪，核心思想就是找到一个可以压的状态。

数位dp:有记忆化和dp两种，一般来说记忆化较为好写好调。同时记忆化后一般从高位向低位转（dp从低位向高位转很麻烦，从高位向低位状态设计和调试都不容易），同时一定要注意前导0，若非要dp，例如AC自动机上的数位dp，可以把是否触界设进状态里面去。

long long dfs(int now,int sum,int lim)

{

if(now<1)return 边界;

if(!lim&&vis[now][sum]!=-1)return vis[now][sum];

int t=lim?z[now]:9;//判断是否压边界

long long tmp=0;//存当前值

for(int i=1;i<=t;i++)

{

//进行转移

tmp+=dfs(now-1,k,lim&&i==t);

}

if(!lim)vis[now][sum]=tmp;

return tmp;

}

AC自动机和kmp上的转移：AC自动机和kmp都是在确定转移关系而已。

斜率优化dp:

往往dp方程有形如f[i]\*g[j]的东西。

标准套路：把要求项化成截距，与要求项相关的化成斜率，然后看已知项的单调性，判断是怎样向凸包中加点：若x单调易做，否则用set动态维护，或者CDQ,即让x天然有序，里面分时间，递归左边，然后直接把左边的凸包建出来，右边算答案即在凸包上贪即可，递归右边，再按x合并即可。这类题有一个细节要注意：当通过乘法判斜率时，要注意是否爆long long,当用除法判断时，注意精度（注意的方法就是不用三态函数，不要写>=，用>即可）。同时当斜率不单调时，可以三分也可以拿一个数组存斜率然后lower\_bound()即可。