本周推荐

POJ 1191 棋盘分割

虽然是上古老题了但时至今日仍然在被大家传颂！均方差的trick不容错过，二维区间DP的思路骨骼清奇，而精度问题更是波诡云谲。详见我下方不远处的题解——Zars19

P4556雨天的尾巴

算是线段树合并的模板题，但是也比较综合，把树上差分和线段树合并放在了一起0.0—— _wzx27

SPOJ-SUBLEX

很经典的后缀数组转化！还包含了整体二分的思想，锻炼码力和思维的不二选择 ——Infinity37

队伍训练情况

本周暂无队伍训练

_wzx27

线段树合并

线段树合并

简介

~~线段树合并是怎么回事呢？线段树大家应该很熟悉，那么…~~

有时候会遇到树上每个点开一个桶或者一个线段树，然后在dfs过程中不断合并的情况，一般权值线段树可以替代桶，然后又因为线段树有较好的合并性质，所以就有了线段树合并的说法。

一般来说关键步骤就是**动态开点**和**线段树的合并操作**

因为不可能真的在每个点建一棵完整的线段树（而且一般情况下每个点所代表的权值都只是完整线段树的小部分），所以要动态开点

```
void update(int &id,int l,int r,int pos,int k)
{
    if(!id) id=++tot;  //动态开点
    if(l==r){
        //更新操作
        return ;
    }
```

```
    int mid = (l+r)>>1;
    if(pos<=mid) update(tr[id].lc,l,mid,pos,k);
    else update(tr[id].rc,mid+1,r,pos,k);
    push_up(id);
}
```

然后合并的操作主要基于两棵线段树具有相同的值域所以有相同的结构，那么递归的处理左右子树即可

```
int mergE(int p,int q,int l,int r)
{
    if(!p)return q;if(!q)return p;
    if(l==r){
        // merge_leaf 操作
        return p;
    }
    int mid = (l+r)>>1;
    tr[p].lc = mergE(tr[p].lc,tr[q].lc,l,mid);
    tr[p].rc = mergE(tr[p].rc,tr[q].rc,mid+1,r);
    push_up(p);
    return p;
}
```

复杂度

空间复杂度比较要注意（因为涉及数组开多大QAQ□□每次update最多开$logn$个点，所以一般开$mlogn$的数组□m表示update次数□n表示权值线段树的值域上限

~~时间复杂度的话就不会搞子~~，主要开销在mergE操作上，两个树的公共点越多开销越大（这好像没办法算啊），但注意一点是如果在一颗树上合并了$n-1$次那么这个复杂度不会比直接建一棵完整的线段树来得大

例题

P4556雨天的尾巴

给出一棵树□m次操作，每次操作让path<u,v>上的点都发放第k种粮食。所有m次操作完后，求每个点存放最多的粮食的种类是什么。

显然是树上点差分，但是值域很大，可以在每个点建一棵权值线段树，然后合并即可，注意差分在update上有常数4，所以开数组要多乘以4

P3605 [USACO17JAN]Promotion Counting P

一棵带点权的树，求每个点u的子树中有多少个点的点权比u自身大

也是每个点建一个权值线段树，权值线段树维护每个值出现的次数，然后dfs过程中合并完后query一下

P3224 HNOI2012]永无乡

n个点，两个操作，一是连接两个点，二是求某个点和所有和他相连的点的第k小

第k小也是权值线段树干的事儿，连接的操作用并查集维护

CF600E. Lomsat gelral

树上每个点染色c_i（好像树上染色可以用dsu（原来我不会啊，那没事了），求每个点子树出现次数最多的颜色的和（可能多个颜色同时出险次数最多）

emm也是线段树瞎搞搞，最后dfs合并一下

Infinity37

专题

后缀数组复习

题目

很摸，复习了下后缀数组

P4051 后缀数组模板题。首先要把这个字符串做成一个环，所以我们就倍长这个字符串。我们知道所谓n种读法就是分别从1~n开始向后读n个构成的字符串，换句话说就是倍长后的字符串的后i个前缀，所以我们就对倍长后的字符串求一个后缀数组，然后把所有开头位置小于等于n的字符首位+n-1输出出来即可。

SPOJ-SUBLEX 我们知道一个字符串的若干字串是由所有后缀的所有前缀组成的，而如果两个后缀产生了公共前缀，那我们就产生了相容的字串，所以我们要求不同的字串，我们可以先对字符串构建后缀素组，对于sa[i]开头的可以产生不同的字串共有n-sa[i]-height[i]个。这样我们就可以计算互不相同的子串有多少个。至于询问第k个是什么，我们直接二分答案就可以了（注意是整体二分）。

比赛

对不起 摸了（土下座）

但是有在建设wiki

Zars19

题目

数位DP想让我告白[1]

开始做一些DP↑ 啊直接外链是不是不太好但我这周摸了确实也没有很多可以写

Update:好吧不好不可以，我搬运一下内容

POJ 1191 二维区间DP

8×8矩阵切$n-1$次成n块（每切一次挑其中一半继续切），问最小均方差

题解：均方差的一种形式是$\sqrt{\frac{\sum{x_i^2}}{n}-{\bar{x}}^2}$平均值其实是固定的，所以最小化$\sum{x_i^2}$二维区间dp

有精度问题要用c++提交才能过g++不行。。恐怖

POJ 3280 区间DP

每个字母有给定的删除\增添代价，问使得字符串变为回文串的最小花费。

题解：这个题是这样，当看到区间dp的那一刻就会了。区间长度从小到大枚举，讨论一下增删，以及左端右端字母本就一致的情况

POJ 2948 DP

有横向运输和纵向运输的两种矿（要送到左端/上方），每个格子只能建一种方向（横/纵）的管道，最大化送到的矿的价值。

题解：想象一下对于每一格子来说如果它建横向那它左边必然都是横向才有意义，纵向同理。搞一下前缀和，状态转移`dp[i][j]=max(dp[i-1][j]+prea[i][j]-prea[i-1][j],dp[i][j-1]+preb[i][j]-preb[i][j-1])`

POJ 2411 状压DP

在$h\times w$的矩阵中铺1×2或2×1的方块，要铺满，问方案数。

题解：看到状压tag了。我的状压是用1表示这一块是2×1的上面那块，需要向下延伸，0表示其他。但是这个样子其实是有一点容易TLE做了一下读入、`if(!f[i-1][k])continue`和`if(h*w%2)puts("0")`的优化卡过去了。看到很多人用记忆化搜索感觉那个样子可能会好一些

POJ 3034 DP

打地鼠游戏，反正长得还挺dp的。给出不同时刻出现在不同坐标的众地鼠，每一时刻到下一时刻只能直线移动，并且击中途径的所有地鼠，移动距离不超过d问最多打中多少地鼠

题解：处理一个不同时刻的地鼠信息数组出来`bool mp[11][40][40]`二重枚举坐标从一时刻转移到下一时刻。值得注意的是*坐标应当被扩大因为锤子可以移动到坐标外*

POJ 2057 树DP

蜗牛把自己的壳丢在某个叶子节点上了于是从根开始找，一些节点上有虫虫可以告诉它的房子在不在这个子树上这样它就可以及时折返节省时间。丢在每个叶子节点的概率相等，某种策略可以使得找到房子的行进路径长度期望最小，问这个期望。

题解：树dp做完一查题解人家的标题都是“ 贪心在动态规划中的应用 ”诶…“ 本题的难点在于如何抉择遍历子节点的顺序 ”呃…我是有什么误解。

我一开始是莽过去的，子节点顺序直接next_permutation了一圈。8的阶乘也才40320诶...n只有1000所以测评姬跑得快不太有问题，不…不好吗。~~不好。~~

$f[u]$表示找到房子的步数之和，$mxlen[u]$表示没找到房子要在这棵子树走的路径长度，$sz[u]$表示子树中的叶子节点个数。个中关系推一推总能得到

有一点理想的话还是研究一下贪心怎么贪。其实也是还挺显而易见的只要用比较优美的形式去表示一下它们之间关系那么

$mxlen[u]=\sum\limits_{v\in{son_u}}(mxlen[v]+2)$（需要往返每个节点）

$f[u]=\sum\limits_{v\in{son_u}}(f[v]+sz[v]+\sum\limits_{w\in{son_u}且在v前}(mxlen[w]+2)\times sz[v])$

所以想象一下排序的过程，两个子节点x,y，x在前意味着多了$(mxlen[x]+2)*sz[y]$，而y在前意味着多了$(mxlen[y]+2)*sz[x]$，故以此为据排序。

POJ 2486 树DP\背包

给定一棵树，从根节点1出发，每个节点上有一定数量苹果，问k步最多摘到多少苹果。

好像是传说中的树dp入门来着，但我竟然，，，没做过。

题解：类似背包。状态表示的关键是节点i走j步所能摘得的最多苹果$f[i][j][l]$的l用来表示是否回到该节点，这里我们用0表示会回到。

转移方程

```
f[u][j][0]=max(f[u][j][0],f[u][l][0]+f[v][j-l-2][0]);
f[u][j][1]=max(f[u][j][1],f[u][l][0]+f[v][j-l-1][1]);
f[u][j][1]=max(f[u][j][1],f[u][l][1]+f[v][j-l-2][0]);
```

每处理一个v时j的循环从大到小可以使得v不产生自我影响

POJ 1947 树DP

给定一棵树，问最少砍断几条边才能得到一个p个节点的子树。

题解：写了一些略显奇怪的东西，随便交居然过了~~不禁让我怀疑数据是不是太水了~~

$f[i][j]$表示节点i子树下保留j个点所需要减掉的最少边数。

这个写法有点形似上一题。对于每个节点u，先递归地把它的子树处理好，之后开始计算$f[u]$数组。没有开始遍历儿子时只初始化$f[u][1]=0$因为一刀也不砍即可保有1个节点即它本身。之后对于每一个儿子v，如果砍掉则$f[u][j]=f[u][j]+1$而$f[u][j]$又可以以$f[u][j-k]+f[v][k]$更新。

由于是假定根节点为1去遍历的所以答案其实是$\min\{f[1][p],f[2][p]+1,f[3][p]+1,\ldots,f[n][p]+1\}$

比赛

暂无。