树的经典算法

陈锋

sukhoeing@gmail.com, QQ(23543620)

201810

倍增算法

数组A，长度N，若干次询问，每次给定T($0 \le T \le \sum_{i=1}^{N} A_i$)。求出满足$\sum_{i=1}^{k} A_i \le T$的最大的k。必须在线回答。

【分析】

- 朴素做法：从前到后枚举k，每次询问最坏O(n)
- 预处理前缀和S，然后在S[1…N]中二分即可，upper_bound
  - 如果T很小，就浪费多一些时间
- 倍增算法
  - 通过若干次2^i的区间拼接成答案
  - 答案为K，O(logK)

```
int p = 1, k = 0, sum = 0;
while(p){ // sum = ∑A[1...k] ≤ T
    int sp = S[k+p] - S[k];
    if(sum + sp <= T) sum += sp, k += p, p *= 2;
    else p /= 2;
}
return k;
```

ACM-ICPC 北京 (2016)网络赛 Genius ACM

给定一个整数M以及整数集合S，定义校验值V(S)如下：

从S中取出最多M对数(2*M个数，不能重复取，不够就别取了)。记这些数为$a_1,b_1,a_2,b_2\cdots\cdots,a_M,b_M$,则要求$\sum_i^M(a_i-b_i)^2$最大，这个最大值就是V(S)。

给一个长度N的数组A以及整数T。把A分成若干段，每段的V值≤T。求最少需要分成几段?

【分析】

1. 对于S，显然该取最大和最小的M个数，第1大&小一对，第2大&小一对……
2. 从左到右，分的每一段尽量包含多的数。分完的段数就是答案
3. 问题转化为，给定L，在满足V(A[L⋯R])≤T的前提下，求R最大值。
4. 求长度为a区间的V值需要O(aloga)。如果使用二分，需要在[L,N]中二分R，上来就要求长度 (N-2)/L区间的V值，但是求出的R可能距离L不远。
5. 求R使用上题的倍增法，对每个L来说上述过程最多循环logN次，每次排序的区间长度之和为NlogN。所以总复杂度为$Nlog^2N$。

倍增计算LCA(Lowest Common Ancestor)

设G是一棵树。对于形式如(u,v)的每个查询，找到节点u和v的最低共同祖先，换句话说，期望的节点w是u和v的CA。特别地，如果u是v的祖先，则u是它们的LCA。

【分析】

1. 对于每个结点i，预处理并记录他的2^j级祖先up(i,j)，j=0…ceil(log(N))，其中up(i,0)是i的父亲。这样从i可以用logN的时间到达它的任意级祖先。可以使用一次DFS来计算up。
2. 对于每个u，我们记住第一次DFS访问u的时间T1(u)，以及离开u的时间T2(u)。利用此信息可以常数时间确定u是不是另一个v的祖先。
   1. $T1(u)<T1(v)<T2(v)<T2(u)$就说明u是v的祖先。
3. 对于查询LCA(u,v)，不妨设dep(u) $<$ dep(v)，否则交换u,v。如果u是v的祖先，直接返回u即可。否则继续。
4. 先沿着u往上爬，爬到最高且不是v祖先的节点x(x不是v的祖先，但是up(x,0)是)。使用倍增算法O(logN)即可找到。

HDU2586

```
// LCA
void dfs(int u, int fa) {
  Tin[u] = ++timer, UP[u][0] = fa;
  for(int i = 1; i < L; i++)
    UP[u][i] = UP[UP[u][i-1]][i-1];
  _for(i, 0, G[u].size()){
    const auto& e = G[u][i];
    if(e.v != fa) Dist[e.v] = Dist[u] + e.k, dfs(e.v, u);
  }
  Tout[u] = ++timer;
}

bool isAncestor(int u, int v) { return Tin[u] <= Tin[v] && Tout[u] >= Tout[v]; }

int LCA(int u, int v){
  if(isAncestor(u, v)) return u;
  if(isAncestor(v, u)) return v;
  for(int i = L; i >= 0; --i) if(!isAncestor(UP[u][i], v)) u = UP[u][i];
  return UP[u][0];
}
```

例题21 邦德(Bond,Uva 11354)

有n座城市通过m条双向道路相连($2 \leqslant n \leqslant 50000$，$1 \leqslant m \leqslant 100000$)，每条道路都有一个危险系数d($0 \leqslant d \leqslant 10^9$)。你的任务是回答Q($1 \leqslant Q \leqslant 50000$)个询问，每个询问包含一个起点s和一个终点t，要求找到一条从s到t的路，使得途径所有边的最大危险系数最小。输出最优路线上所有边的危险系数的最大值。

【分析】

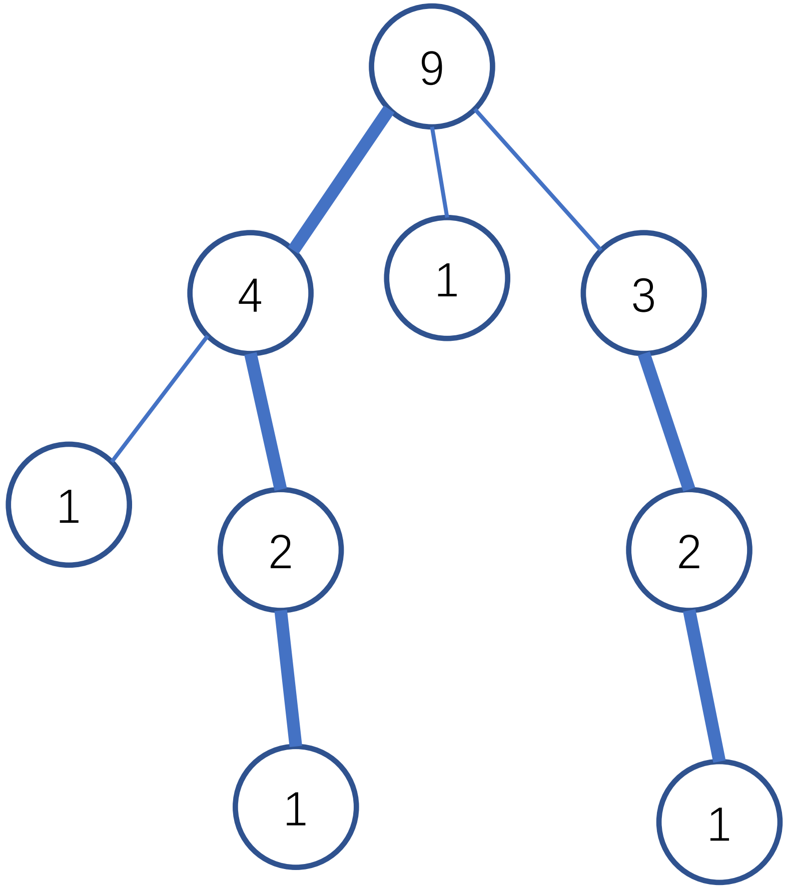
1. 先求出MST，O(MlogM)，也就是最小瓶颈树(最大边权值尽量小)。
2. 对于任意点对(s,t)，首先求出l = LCA(s,t)，但是最大w怎么办?
3. 记录MaxW[i,j]→i到其第j级祖先的路上的最大权值。
4. 使用倍增逻辑计算maxw(u,v)，其中u是v的祖先。
5. 答案就是max(maxw(l, u), maxw(l ,v))。LCA预处理:NLogN。
6. Q次查询：QlogN。

作业

- HDU2586 LCA
- POJ 3417 Network
- https://www.codechef.com/problems/DRAGONST

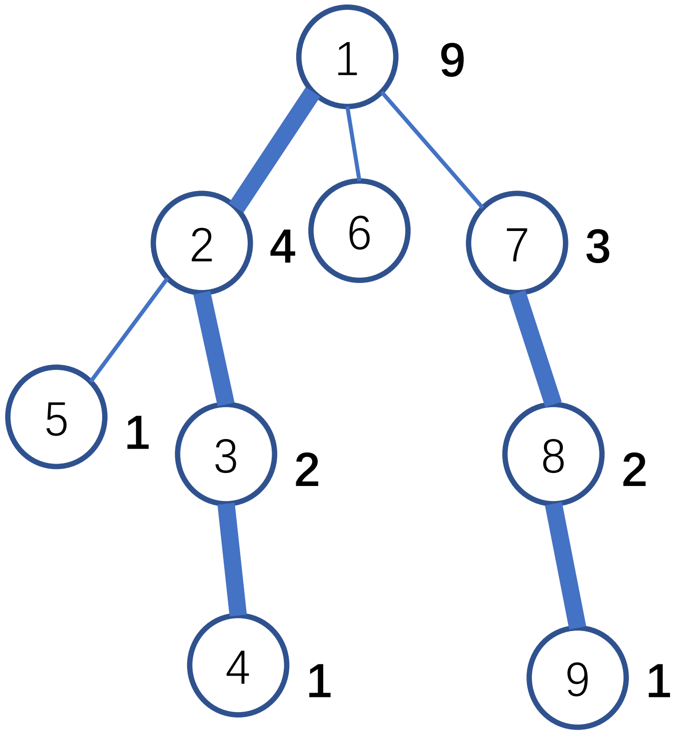
轻重边

- 给定一棵有根树T，对于每个非叶结点u，设u的子树中结点数最多的子树的树根为v
- 标记(u,v)为重边，(u,*)均为轻边.
- 有重复则任取一个，任意结点下面重边只有1个重边(heavy edge)



轻重剖分

- 一次DFS能把T分解成若干重路径(全部由重边组成的路径，有些称为树链)和若干轻边。轻重路径剖分也称树链剖分：
- 以root为起点，沿重边向下拓展，拉成重链。
- 不在当前重链上的，都以该节点为起点向下重新拉一条重链
- tid(x)为新的编号(圈中)，其实就是DFS序的编号。沿重边向下扩展，剖分完成后，一条重链上的结点新编号会是连续的
- 记top(x)为x所在重链的祖先



x(tid)	1	2	3	4	5	6	7	8	9
top[x]	1	1	1	1	5	6	7	7	7

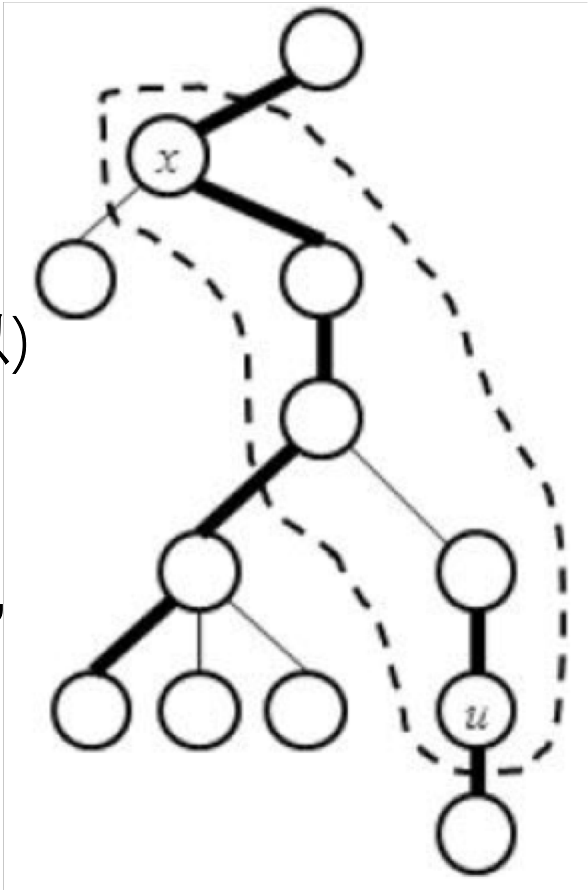
```
label = 1
dfs(x, ancestor):
  tid[x] = label++
  top[x]= ansestor
  dfs(v, ancestor) // u-v is heavy
  for v in u.children:
     dfs(v, v) // u-v light
```

路径剖分性质

- **重要：**若v是u的子结点，(u,v)是轻边，则$size(v)<size(u)/2$，其中size(u)表示以u子树中的结点数。
- 对于任意非root结点u，在u到root的路径上，轻边和重路径的条数均不超过$\log_2 n$。
- root到u的路径上，轻边最多的情况，就是所有的边都是轻边，那么不可能超过$O(\log_2 n)$条，因为每碰到一条轻边，size就会减半。
- 或者是一条重路径与一条轻边交替出现，所以重路径的条数最多和轻边条数一样，也就是不超过$O(\log_2 n)$条。

树的动态查询问题

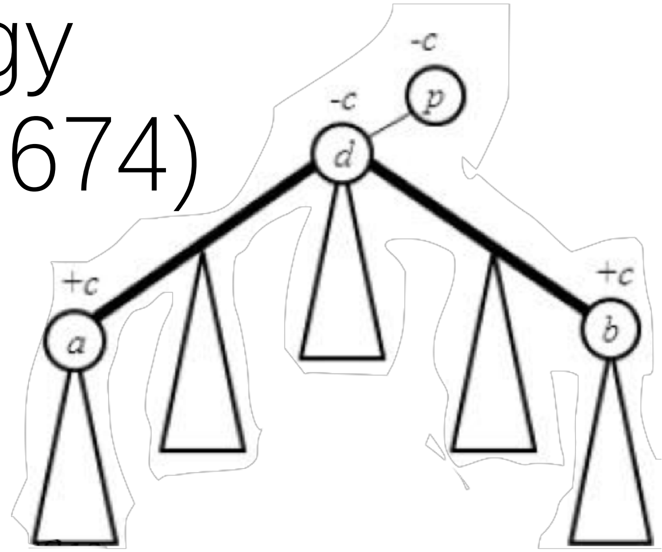
- 给定一棵带边权的树T，要求支持两种操作：
  1. 修改某条边的权值
  2. 询问树中某两点的唯一路径上最大边权(sum,min,max类似)
- 把无根树变成有根树并且求出路径剖分，如图所示
- 任意u到其祖先x的**简单路径**中包含一些轻边和重路径，但这些重路径可能并不是原树中的完整重路径，而只是一些“片段”，因此可以在轻边中直接保存边权，而用线段树维护重路径(A[i..j]对应tid的i…j,A[u]就是u到其父亲的边权)。这样，两个操作都不难实现。



树的动态查询问题

- 修改：轻边直接修改，重边需要在重路径对应的线段树中修改。
- 查询：设LCA(u,v)=p，则只需求出u到p之间的最大边权MW(u,p)，再用求出MW(v,p)，则答案为max{MW(u,p), MW(v,p)}。
- 如何求MW(u,p)，依次访问u到p之间的每条重路径和轻边即可。根据刚才的结论，轻边和重路径的条数均不超过$\log_2 n$。
- 修改的时间为O(logn)，查询为$O(\log_2 n)$。
  - 存在时间复杂度更低的方法(多个线段树可以合并成1个)，但上述方法已经很实用了。

例题12-6 闪电的能量(Lightning Energy Report, ACM/ICPC Jakarta 2010, UVa1674)



有n(n≤50000)座房子形成树，还有Q(Q≤10000)道闪电。每次闪电会打到两个房子a,b，你需要把二者路径上的所有点(包括a,b)的闪电值加上c(c≤100)。最后输出每个房子的总闪电值。

【分析】

1. 如果输入是个数组而不是树，那么就是裸的线段树。因为是树，所以需要把树切成一段段的数组。问题是如何切分才足够快。轻重剖分正好，可以把树切成≤lg(n)段。
2. 标准解法是利用路径剖分：每次最多更新2logn条重路径，而每条重路径上的区间更新需要O(logn)时间。
3. 定义mark数组，mark[u]=w意思是u到root的路径上每个点的权值增加w。
4. 对于操作(a,b,c)，先求d=LCA(a,b), mark[a]+=c, mark[b]+=c, mark[d]-=c，即i的闪电值等于子树i的总mark值。如果d≠root，还要让d的父结点p的mark[p]-=c。
5. 经过上述mark修改操作之后，只有a到b路径上所有点的“子树总mark值”增加了c，其他结点保持不变。最后用一次DFS，即可求出以每个结点为根的子树的总mark值。

Review-树的重心

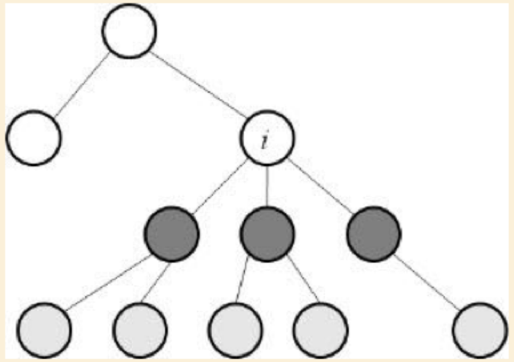


图9-12 结点i的gs(i)(浅灰色)和s(i)(深灰色)

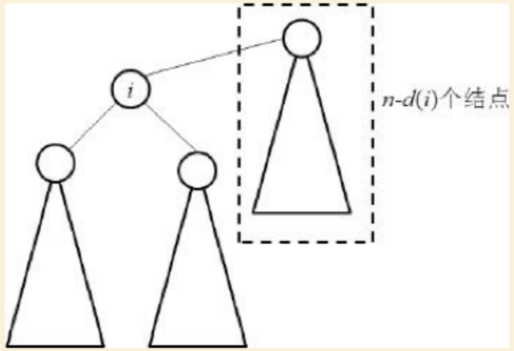


图9-13 树中的结点分布

d(u)为u子树的结点数

d(u) = Σd(v) + 1

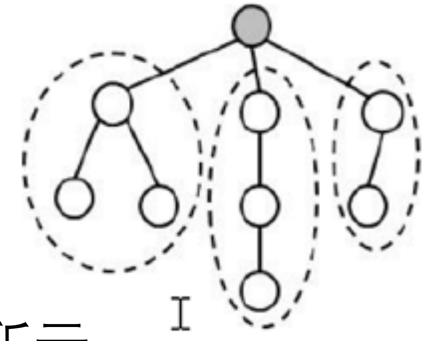
f(u) = max(d(v), u父子树个数=n-d(u))

u = argmin(f(u))

路径统计(点分治, POJ1741)

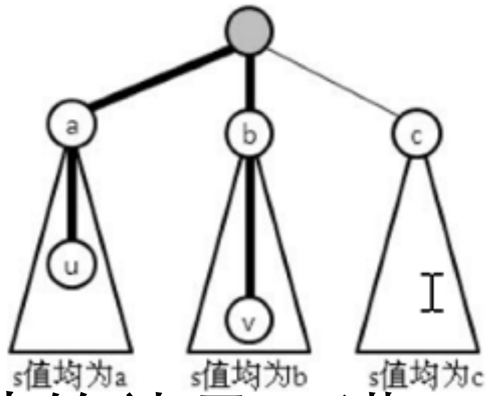
给定一棵n个结点的正权树，定义dist(u,v)为u,v两点间唯一路径的长度(即所有边的权和)，再给定一个正数K，统计有多少对结点(a,b)满足dist(a,b)≤K

【分析】



1. 直接计算任意两个结点之间的距离，O(n^2)。
2. 路径要么经过root，要么完全在一棵子树中
3. 分治：选root将无根树转为有根树，递归处理每一棵子树。如图所示。
4. DP中介绍过树的重心，如果选重心为root，子树的结点个数≤n/2，递归深度不超过O(logn)。需要统计3类路径：
    1. 完全位于一棵子树内的："递归"部分。
    2. 一个端点是root：统计满足d(i)≤K非root结点i的个数, d(i)为i到root路径长度。
    3. 经过root。这种情况比较复杂，需要继续讨论。
5. 递归到子树时，要给子树重新选重心

经过root的路径个数



- 记s(i)表示root的哪棵子树包含i，要统计的就是：满足d(i)+d(j)≤K且s(i)≠(j)的(i,j)个数，如图所示。
- 任意s(i)≠(j)的(i,j)之间都是一条经过root的路径，使用补集转换：
- 设A为满足d(i)+d(j)≤K的(i,j)个数，B为满足d(i)+d(j)≤K且s(i)=s(j)的(i,j)个数，则答案等于A-B。
- A的计算：所有d值排序，进行一次线性扫描。
- B的计算，只不过是对于root的每个子结点分别处理，把s值等于该子结点的所有d值排序，然后线性扫描。
- 根据主定理，总时间复杂度为$O(n(\log n)^2)$。

有序数组上的双指针法

给定一个递增序的数组A，如何O(n)求出满足$A_i+A_j \leqslant K$的点对(i,j)数量？有一个很经典的方法叫“双指针法”，可以高效地解决很多有序数组上的统计任务。

很容易看出这个算法O(n)的，因为每个元素只被访问了一次。正确性来自有序数组的单调性。

```
L = 0, R = |A|-1
while(L<R):
  if A_L+A_R>K:
    R = R-1
  else
    ans += R-L
```

例题12-4 铁人比赛(Ironman Race in Treeland, ACM/ICPC Kuala Lumpur 2008, UVa12161)

给定一棵n个结点的树，每条边包含长度L和费用D(1≤D,L≤1000)两个权值。要求选择一条总费用不超过m的路径，使得路径总长度尽量大。输入保证有解，1≤n≤30000，1≤m≤10^8。

【分析】

- 点分治，关键点是计算经过root的最优路径。DFS求出子树内所有结点到根的路径长度和费用，然后按照DFS序从小到大枚举这些结点。
- 枚举到点i时，记i到root路径的费用为c(i)，长度为d(i)，则需要在i之前的结点(已经枚举过的)中找一个费用不超过m-c(i)的且到root距离最大的结点u。
  - 对于结点u,v，如果c(u)>c(v)但d(u)≤d(v)，则u一定不是最优解的端点，可以删除
- i之前的结点可以组织成单调集合：到根的路径长度和路径费用同时递增。如果把这个单调集合保存到BST中，O(logn) 找到“费用不超过给定值的前提下距离最大的结点”。这O(nlogn)求出了“经过root的最优路径”。根据主定理，总时间为O(n(logn)^2)。
- 子树递归求解之前，每次重新计算重心