倍增思想的应用

徐毅

江苏省常州高级中学

August 13, 2014

What

倍增思想是什么呢？

What

倍增思想是什么呢？
倍增，顾名思义，就是每次增加一倍。

What

倍增思想是什么呢？

倍增，顾名思义，就是每次增加一倍。

展开来说，就是每次根据已经得到的信息，将考虑的范围增加一倍，从而加速操作。

Why

倍增思想有什么用呢？

Why

倍增思想有什么用呢？

这是一种非常巧妙的思想，可以用来解决信息学竞赛中的很多问题，一个经典应用是后缀数组的构造。

How

倍增思想该怎么用呢？

How

倍增思想该怎么用呢？

考虑这样一个比较一般的模型，在一个有向图中，每个点最多只有一条出边，每条边有一定的信息，走过一条路径时，就将路径上边的信息依次按一定的规则合并，并且合并的规则满足结合律。

How

那么，对于点 x，我们可以倍增地得到它经过 2^i 条边后到达的点 $next(x,i)=next(next(x,i-1),i-1)$，并维护路径上边的信息合并后的结果 $info(x,i)=merge(info(x,i-1),info(next(x,i-1),i-1))$。

How

那么，对于点 x，我们可以倍增地得到它经过 2^i 条边后到达的点 $next(x, i) = next(next(x, i-1), i-1)$，并维护路径上边的信息合并后的结果 $info(x, i) = merge(info(x, i-1), info(next(x, i-1), i-1))$。

预处理出上面的内容后，我们就要回答有关这个图的询问，这就是我们接下来讨论的主要内容。

Example 1

求 $x^y \mod p(1 \leq x, y, p \leq 10^9)$。

Example 1

快速幂这个算法大家应该是耳熟能详了，古老的写法是折半递归：

Example 1

快速幂这个算法大家应该是耳熟能详了，古老的写法是折半递归：

```
int pow(int x, int y) {
    if (!y)
        return 1;
    int t = pow(x, y >> 1);
    t = (long long)t * t % p;
    return y & 1 ? (long long)t * x % p : t;
}
```

Example 1

现在更常见的是非递归写法，也是倍增思想应用的经典例子：

Example 1

现在更常见的是非递归写法，也是倍增思想应用的经典例子：

```
int pow(int x, int y) {
    int t = 1;
    for (; y; y >>= 1) {
        if (y & 1)
            t = (long long)t * x % p;
        x = (long long)x * x % p;
    }
    return t;
}
```

Example 1

这个问题和我们之前提到的模型有什么关系呢?

Example 1

这个问题和我们之前提到的模型有什么关系呢？

把这个问题表示成 $x^0 \to x^1 \to x^2 \to \cdots \to x^y$ 这样一个图，图中的边权都是 x，合并的规则是乘法。

Example 1

最坏有 10^9 个点，之前提到的预处理怎么办呢？

Example 1

最坏有 10^9 个点，之前提到的预处理怎么办呢？

我们要求的是 $x^0 \to x^y$ 路径的边权积。而这个图非常特殊，所有的边权相同，那么路径的边权积就只和路径的边数有关，并不需要关心从哪个点到哪个点。

Example 1

最坏有 10^9 个点，之前提到的预处理怎么办呢？

我们要求的是 $x^0 \to x^y$ 路径的边权积。而这个图非常特殊，所有的边权相同，那么路径的边权积就只和路径的边数有关，并不需要关心从哪个点到哪个点。

因此，我们只要预处理 2^i 条边的边权积就可以了。

Example 1

答案即 y 条边的边权积怎么求呢？

Example 1

答案即 y 条边的边权积怎么求呢?

利用预处理结果回答询问的核心手段是二进制拆分，也就是把一个数拆分成若干个 2 的幂的和。

Example 1

答案即 y 条边的边权积怎么求呢?

利用预处理结果回答询问的核心手段是二进制拆分，也就是把一个数拆分成若干个 2 的幂的和。

那么在这里，例如我们要求 5 条边的边权积，$(5)_{10} = (101)_2$，就只要把 2^2 条边的边权积和 2^0 条边的边权积相乘就可以了。

Example 1

答案即 y 条边的边权积怎么求呢？

利用预处理结果回答询问的核心手段是二进制拆分，也就是把一个数拆分成若干个 2 的幂的和。

那么在这里，例如我们要求 5 条边的边权积，$(5)_{10} = (101)_2$，就只要把 2^2 条边的边权积和 2^0 条边的边权积相乘就可以了。

时间复杂度 $O(\log y)$。

Example 2

求 $x^y \mod p(1 \leq x, y, p \leq 10^{18})$。

Example 2

看起来和上一题没什么区别？

Example 2

看起来和上一题没什么区别？
只要把所有数都改成 long long 类型就可以了？

Example 2

看起来和上一题没什么区别？
只要把所有数都改成 long long 类型就可以了？
$10^{18} \times 10^{18} =?$

Example 2

高精度乘法?

Example 2

高精度乘法？
用两个 long long 类型拼起来模拟一个 128 位整数。

Example 2

求 $xy \mod p$ 有没有更无脑的方法?

Example 2

求 $xy \mod p$ 有没有更无脑的方法?

把这个问题表示成 $0x \to 1x \to 2x \to \cdots \to yx$ 这样一个图，图中的边权都是 x，合并的规则是加法。

Example 2

似曾相识?

Example 2

似曾相识?

```
long long mul(long long x, long long y) {
    long long t = 0;
    for (; y; y >>= 1) {
        if (y & 1)
            t = (t + x) % p;
        x = x << 1 % p;
    }
    return t;
}
```

这就是所谓的快速乘。

Example 2

```
long long pow(long long x, long long y) {
    long long t = 1;
    for (; y; y >>= 1) {
        if (y & 1)
            t = mul(t, x);
        x = mul(x, x);
    }
    return t;
}
```

时间复杂度 $O(\log p \log y)$。

Example 3: 转圈游戏[1]

$n(1 \le n \le 10^6)$ 个人围成一圈，位置顺时针编号为 $0 \sim n-1$。

在每一轮中，所有人同时行动，位置 i 上的人将走到位置 $(i+m) \mod n (0 \le m < n)$ 上。

求初始在位置 x 上的人 $10^k (0 \le k \le 10^9)$ 轮后走到了哪个位置上。

[1]Source: NOIP 2013 提高组

Example 3: 转圈游戏

答案显然为 $(x+10^k m) \mod n$，使用快速幂即可。
时间复杂度 $O(\log k)$。

Example 4: 容易题[2]

长度为 $m(1 \le m \le 10^9)$ 的数列 a 各项均为 $1 \sim n(1 \le n \le 10^9)$ 的整数，同时要满足 $k(0 \le k \le 10^5)$ 条形如 (i, x) 的限制表示 $a_i \ne x$。

求所有可能的数列 a 的各项乘积的和 $\mod 1000000007$。

[2]Source: HAOI 2012

Example 4: 容易题

答案显然为每一项能取的数的和的乘积。

设 t 为受限制的项数，我们可以首先把受限制的项能取的数的和的乘积求出来，再乘上 $(\frac{n(n+1)}{2})^{m-t}$。

Example 4: 容易题

答案显然为每一项能取的数的和的乘积。

设 t 为受限制的项数，我们可以首先把受限制的项能取的数的和的乘积求出来，再乘上 $(\frac{n(n+1)}{2})^{m-t}$。

前者排序后扫一遍即可，后者可以用快速幂完成。

时间复杂度 $O(k \log k + \log m)$。

Transition

除了在快速幂中使用的二进制拆分之外，还有一些特殊的问题可以使用其他回答询问的方法。

Example 5

给出长度为 $n(1 \leq n \leq 10^5)$ 的序列 a 和 $q(1 \leq q \leq 10^7)$ 组形如 (l, r) 的询问，每次询问 $\min_{i=l}^{r} a_i$。

Example 5

联系前面的模型，第 i 项指向第 $i+1$ 项，边权是 a_i，合并的规则是取 min。

Example 5

联系前面的模型，第 i 项指向第 $i+1$ 项，边权是 a_i，合并的规则是取 min。

```
for (int i = 1; i <= n; ++i) {
    lg[i] = lg[i - 1] + (1 << lg[i - 1] + 1 == i);
    info[i][0] = a[i];
}
for (int k = 1; k <= lg[n]; ++k)
    for (int i = 1; i + (1 << k) - 1 <= n; ++i)
        info[i][k] = min(info[i][k - 1],
                         info[i + (1 << k - 1)][k - 1]);
```

Example 5

仍然使用二进制拆分回答询问。

Example 5

仍然使用二进制拆分回答询问。
时间复杂度 $O((n+q)\log n)$？

Example 5

取 min 的特殊性？

Example 5

取 min 的特殊性？

有重叠部分仍然可以合并！对于询问 (l, r)，令 $k = \lfloor \log_2(r - l + 1) \rfloor$，则答案为 $\min\{info(l, k), info(r - 2^k + 1, k)\}$。

Example 5

取 min 的特殊性？
有重叠部分仍然可以合并！对于询问 (l, r)，令
$k = \lfloor \log_2(r - l + 1) \rfloor$，则答案为 $\min\{info(l, k), info(r - 2^k + 1, k)\}$。
时间复杂度 $O(n \log n + q)$。

Transition

还有一些问题，并不是给定一个量让你去拆分，而是需要你去寻求这个量的极限值。

Example 6: 开车旅行[3]

$n(1 \le n \le 10^5)$ 个城市在一直线上，自西向东编号为 $1 \sim n$，且有互不相同的海拔高度。两城市间距离为他们海拔高度差的绝对值。

A, B 两人轮流开车，A 先开，之后每天轮换。他们从起点 s 出发一直向东开，B 每次沿前进方向选最近的城市作为目的地，A 则选第二近的（距离相同时离海拔低的更近），当其中一人无法选择目的地或到达目的地将会使总距离超过 x 则停止。

给出 $m(1 \le m \le 10000)$ 组形如 (s, x) 的询问，每次询问 A 和 B 分别开的距离。

[3]Source: NOIP 2012 提高组

Example 6: 开车旅行

首先要知道 A, B 从每个城市出发选择的目的地?

Example 6: 开车旅行

首先要知道 A, B 从每个城市出发选择的目的地？

对于一个城市，考虑它与它东面的所有城市的海拔高度，离它最近的城市一定是它的前驱或者后继。

- 如果是前驱，则离它第二近的城市一定是它前驱的前驱或者它的后继；
- 如果是后继，则离它第二近的城市一定是它后继的后继或者它的前驱。

Example 6: 开车旅行

首先要知道 A, B 从每个城市出发选择的目的地？

对于一个城市，考虑它与它东面的所有城市的海拔高度，离它最近的城市一定是它的前驱或者后继。

- 如果是前驱，则离它第二近的城市一定是它前驱的前驱或者它的后继；
- 如果是后继，则离它第二近的城市一定是它后继的后继或者它的前驱。

那么，直接倒过来求，用 set 来维护？

Example 6: 开车旅行

更简单的方法?

Example 6: 开车旅行

更简单的方法？

把所有城市按海拔高度排序，就得到了所有城市的前驱后继关系。正过来求，每处理完一个城市就将其删除，即让它的前驱和它的后继相连。

Example 6: 开车旅行

更简单的方法?

把所有城市按海拔高度排序，就得到了所有城市的前驱后继关系。正过来求，每处理完一个城市就将其删除，即让它的前驱和它的后继相连。

用双向链表即可。

Example 6: 开车旅行

接下来，我们发现这个图就和模型基本吻合了，边权就是距离，合并的规则是加法，只不过 A 和 B 要分开算。

Example 6: 开车旅行

接下来，我们发现这个图就和模型基本吻合了，边权就是距离，合并的规则是加法，只不过 A 和 B 要分开算。

怎么知道何时停止呢？

Example 6: 开车旅行

接下来，我们发现这个图就和模型基本吻合了，边权就是距离，合并的规则是加法，只不过 A 和 B 要分开算。

怎么知道何时停止呢？

边数未知，还能二进制拆分吗？

Example 6: 开车旅行

从二进制高位向低位确定!

Example 6: 开车旅行

从二进制高位向低位确定！

我们将 i 从 16 枚举到 0，如果已经开的距离加上往前开 2^i 天的距离不超过 x，就可以往前开 2^i 天，也就确定了极限边数二进制中第 i 位为 1。

到达极限后，此时 A 和 B 开的距离就是答案。

Example 6: 开车旅行

从二进制高位向低位确定！

我们将 i 从 16 枚举到 0，如果已经开的距离加上往前开 2^i 天的距离不超过 x，就可以往前开 2^i 天，也就确定了极限边数二进制中第 i 位为 1。

到达极限后，此时 A 和 B 开的距离就是答案。

时间复杂度 $O((m+n)\log n)$。

Transition

看了这么多例子，相信大家对倍增思想的应用已经有了初步的认识。

而倍增思想应用最广泛、最灵活的就是树上的问题了。

Example 7

给出一棵 $n(1 \le n \le 10^5)$ 个点的有根树和 $q(1 \le q \le 10^5)$ 组形如 (x, y) 的询问，每次询问 $LCA(x, y)$ 即 x 和 y 的最近公共祖先。

Example 7

树就是模型中的图？不过在这里我们不关心边权。

Example 7

树就是模型中的图？不过在这里我们不关心边权。

在 DFS 或 BFS 遍历这棵树的过程中，走到点 x 时我们就可以完成对点 x 的预处理。

```
for (int k = 1; k <= lg[dep[x]]; ++k)
    nxt[x][k] = nxt[nxt[x][k - 1]][k - 1];
```

Example 7

怎么回答询问呢？一起往上爬？

Example 7

怎么回答询问呢？一起往上爬？
x 和 y 深度不同？

Example 7

怎么回答询问呢？一起往上爬？
x 和 y 深度不同？
先爬到同一深度！

Example 7

怎么回答询问呢？一起往上爬？

x 和 y 深度不同？

先爬到同一深度！

不妨设 $dep(x) > dep(y)$，我们只要对 $dep(x) - dep(y)$ 进行二进制拆分，就可以使 x 爬到 y 的深度。

Example 7

怎么回答询问呢？一起往上爬？

x 和 y 深度不同？

先爬到同一深度！

不妨设 $dep(x) > dep(y)$，我们只要对 $dep(x) - dep(y)$ 进行二进制拆分，就可以使 x 爬到 y 的深度。

已经到同一个点了？完工！

Example 7

否则就可以一起往上爬了？

Example 7

否则就可以一起往上爬了？

我们要求的是 x 和 y 一起往上爬且不到达同一个点经过的极限边数。似曾相识？

Example 7

否则就可以一起往上爬了？

我们要求的是 x 和 y 一起往上爬且不到达同一个点经过的极限边数。似曾相识？

同样是从二进制高位向低位确定！

Example 7

否则就可以一起往上爬了？

我们要求的是 x 和 y 一起往上爬且不到达同一个点经过的极限边数。似曾相识？

同样是从二进制高位向低位确定！

别忘了经过极限边数到达的点的父亲才是我们要的。

Example 7

```
int lca(int x, int y) {
    if (dep[x] < dep[y])
        swap(x, y);
    while (dep[x] > dep[y])
        x = nxt[x][lg[dep[x] - dep[y]]];
    if (x == y)
        return x;
    for (int k = lg[dep[x]]; k >= 0; --k)
        if (nxt[x][k] != nxt[y][k])
            x = nxt[x][k], y = nxt[y][k];
    return nxt[x][0];
}
```

Example 7

```
int lca(int x, int y) {
    if (dep[x] < dep[y])
        swap(x, y);
    while (dep[x] > dep[y])
        x = nxt[x][lg[dep[x] - dep[y]]];
    if (x == y)
        return x;
    for (int k = lg[dep[x]]; k >= 0; --k)
        if (nxt[x][k] != nxt[y][k])
            x = nxt[x][k], y = nxt[y][k];
    return nxt[x][0];
}
```

时间复杂度 $O((n+q)\log n)$。

Example 8: Query on a tree II[4]

给出一棵 $n(1 \le n \le 10^5)$ 个点的树（边有权）和 $q(1 \le q \le 10^5)$ 组形如 (x, y) 或 (x, y, k) 的询问，(x, y) 表示询问 $x \to y$ 的路径长度，(x, y, k) 表示询问 $x \to y$ 路径上的第 k 个点。

[4]Source: SPOJ QTREE2

Example 8: Query on a tree II

和上一题相同的预处理，只是多了边权，合并的规则是加法。

Example 8: Query on a tree II

和上一题相同的预处理，只是多了边权，合并的规则是加法。
回答询问 (x, y)？

Example 8: Query on a tree II

和上一题相同的预处理，只是多了边权，合并的规则是加法。
回答询问 (x, y)？
$x \to y$ 即 $x \to LCA(x, y) \to y$，只要在求 $LCA(x, y)$ 时顺便把边权和进行累加。

Example 8: Query on a tree II

回答询问 (x, y, k)? 第 k 个点在哪?

Example 8: Query on a tree II

回答询问 (x, y, k)？第 k 个点在哪？

- 当 $k \leq dep(x) - dep(LCA(x, y)) + 1$ 时，第 k 个点在 $x \to LCA(x, y)$ 路径上，即 x 向上爬 $k - 1$ 条边后到达的点；
- 当 $k \geq dep(x) - dep(LCA(x, y)) + 1$ 时，第 k 个点在 $LCA(x, y) \to y$ 路径上，即 y 向上爬 $dep(x) + dep(y) - 2dep(LCA(x, y)) - k + 1$ 条边后到达的点。

Example 8: Query on a tree II

回答询问 (x, y, k)？第 k 个点在哪？

- 当 $k \leq dep(x) - dep(LCA(x, y)) + 1$ 时，第 k 个点在 $x \to LCA(x, y)$ 路径上，即 x 向上爬 $k - 1$ 条边后到达的点；
- 当 $k \geq dep(x) - dep(LCA(x, y)) + 1$ 时，第 k 个点在 $LCA(x, y) \to y$ 路径上，即 y 向上爬 $dep(x) + dep(y) - 2dep(LCA(x, y)) - k + 1$ 条边后到达的点。

向上爬又是二进制拆分的过程了。
时间复杂度 $O((n + q) \log n)$。

Example 9: 疫情控制[5]

给出一棵 $n(2 \le n \le 50000)$ 个点的有根树（边有权），任务是需要一些人将某些点标记为特殊点（根不能被标记），使得从根到每个叶子结点的路径上都有至少一个特殊点。

现在有 $m(2 \le m \le n)$ 个人，初始在各自的点上，在树上移动的速度为 1，最终选择一个点进行标记。

求完成任务的最短时间。

[5] Source: NOIP 2012 提高组

Example 9: 疫情控制

看起来无从下手？

Example 9: 疫情控制

看起来无从下手？
二分答案！
如何验证？

Example 9: 疫情控制

看起来无从下手？
二分答案！
如何验证？
贪心！

Example 9: 疫情控制

首先，一个人在到达根之前一定会往上走，若到达根后还能走才会选择往下一步去控制另一棵子树。

Example 9: 疫情控制

首先，一个人在到达根之前一定会往上走，若到达根后还能走才会选择往下一步去控制另一棵子树。

那么，我们需要和上一题一样的预处理，然后就能用从二进制高位向低位确定的方法得到每个人在时间 t 内能往上到达的最高点，若能到达根还要记录他原本所属的子树以及剩余的时间。

Example 9: 疫情控制

首先，一个人在到达根之前一定会往上走，若到达根后还能走才会选择往下一步去控制另一棵子树。

那么，我们需要和上一题一样的预处理，然后就能用从二进制高位向低位确定的方法得到每个人在时间 t 内能往上到达的最高点，若能到达根还要记录他原本所属的子树以及剩余的时间。

对于不能到达根的人，所能到达的最高点一定是他所标记的点。我们把这些点先标记后，再通过 DFS 就能得出还有哪些子树需要控制。

Example 9: 疫情控制

而对于能到达根但剩余时间不足以再到达原本所属的子树的人，若原本所属的子树还未被控制，就不选择到根返回去控制原本所属的子树。

Example 9: 疫情控制

而对于能到达根但剩余时间不足以再到达原本所属的子树的人，若原本所属的子树还未被控制，就不选择到根返回去控制原本所属的子树。

我们将剩下的人按剩余时间从小到大排序，将根到需要控制的子树的边权也从小到大排序，若每棵子树都能被分配到人说明时间 t 是合法的。

Example 9: 疫情控制

而对于能到达根但剩余时间不足以再到达原本所属的子树的人，若原本所属的子树还未被控制，就不选择到根返回去控制原本所属的子树。

我们将剩下的人按剩余时间从小到大排序，将根到需要控制的子树的边权也从小到大排序，若每棵子树都能被分配到人说明时间 t 是合法的。

时间复杂度 $O((m\log m + (m+n)\log n)\log t)$。

Example 10: 货车运输[6]

给出一个 $n(1 \le n \le 10000)$ 个点 $m(1 \le m \le 50000)$ 条有容量限制边的无向图和 $q(1 \le q \le 30000)$ 组形如 (x, y) 的询问，每次询问货车从 x 到 y 最多能运输多少货物。

[6] Source: NOIP 2013 提高组

Example 10: 货车运输

要让 $x \to y$ 路径的最小边最大?

Example 10: 货车运输

要让 $x \to y$ 路径的最小边最大?
最大生成树!

Example 10: 货车运输

要让 $x \to y$ 路径的最小边最大？

最大生成树！

求出原图的最大生成树后，我们就把问题转化到了树上，每次要查询 $x \to y$ 路径上的最小边。

Example 10: 货车运输

和之前类似的预处理，合并的规则是取 min。
回答询问大家应该也会了。
时间复杂度 $O(m\log m + (n+q)\log n)$。

Summary

从倍增思想应用的一个基本模型出发，就能产生形形色色的问题。

Summary

从倍增思想应用的一个基本模型出发，就能产生形形色色的问题。
举一反三。

Summary

从倍增思想应用的一个基本模型出发，就能产生形形色色的问题。
举一反三。
熟能生巧。

Thanks

谢谢大家。