计算机网络实验报告-09

阮星程
2015K8009929047

一、 实验题目

高效 IP 路由查找实验。

二、 实验内容

理解 IP 查找的内容和原理，构建 IP 查找程序，并满足下列实验要求：

- 实现最基本的前缀树查找。
- 实现多 bit 的前缀树查找，并完成叶推，压缩向量，压缩指针等优化。
- 对比两种查找的时间开销。

三、 实验过程

首先构建基本的前缀树查找程序。

普通前缀查找树节点的结构如下：

```
typedef struct basic_tree_node{
    struct basic_tree_node * son0;
    struct basic_tree_node * son1;
    int matched;         //whether this node is a matched node
    u32 prefix;          //matched: corresponding ip, unmatched: 0
    u32 mask;            //matched: corresponding mask(1 - 32), unmatched: 0
} btn;
```

读入文件并将表项添加到树中。

```
//whole process of making basic tree and matching
btn * basic_prefix_match()
{
    printf("start basic tree build\n");
    FILE * fp = fopen("forwarding-table.txt","r");
    btn * root = btn_init(NULL, NULL, 0);
    u32 ip0, ip1, ip2, ip3, ip;
    u32 mask;
    while(!feof(fp))
    {
        fscanf(fp, "%u.%u.%u.%u %d %d\n", &ip0, &ip1, &ip2, &ip3, &mask, &ip);
        ip = (ip0<<24) + (ip1<<16) + (ip2<<8) + ip3;
        bt_add_node( root, ip, mask, 1);
    }
```

其中的 bt_add_node 函数构建如下：

```
//add node to basic tree
void bt_add_node(btn * root, u32 ip, u32 mask, u32 start)
{
    if(start <= mask)
    {
        int j = ip & (1<<(32 - start));
        if(j)
        {
            if(!root->son1)
                root->son1 = btn_init(NULL, NULL, 0);
            bt_add_node( root->son1, ip, mask, start + 1);
        }
        else
        {
            if(!root->son0)
                root->son0 = btn_init(NULL, NULL, 0);
            bt_add_node( root->son0, ip, mask, start + 1);
        }
    }
    else
    {
        root->matched = 1;
        root->prefix = ip;
        root->mask = mask;
    }
}
```

其中的 start 项输入是为了方便递归构造，指示这一次将从哪个 bit 开始匹配添加。首先判断是否已经匹配了所有的掩码覆盖的项，如是的话，则将对应节点表示为 matched，并对其赋相应的 prefix 和 mask 值；否则就继续匹配添加下一个字节。

树构建完成后可以开始匹配：

```
//use basic tree to match ip
int bt_match(btn * root, u32 ip, u32 mask, u32 start)
{
    if (root->matched)
        last_matched = root;
    if(start <= mask)
    {
        int j = ip & (1<<(32 - start));
        if(j)
        {
            if(!root->son1)
                return root->matched;
            return bt_match( root->son1, ip, mask, start + 1);
        }
        else
        {
            if(!root->son0)
                return root->matched;
            return bt_match( root->son0, ip, mask, start + 1);
        }
    }
    else
    {
        return root->matched;
    }
}
```

基本架构与 bt_add_node 函数一致，只是为了避免回溯，添加了一个 last_matched 的全局变量，这样搜索到头时，如果没有相应匹配，就只需要看这个变量而不再需要回溯，否则对于我们的树结构来说，要回溯是相当麻烦的。

接下来开始一些优化，首先进行叶推，将 btn 树中的中间节点下推到叶子节点，由于之后的架构关系，我们不必要考虑将中间节点的 matched 删去再进行下推，所以可以将叶推构建得很简单。

```
//finish leaf pushing work, change a basic tree
void leaf_pushing(btn * root, u32 prefix, u32 mask)
{
    if(mask > root->mask)
    {
        root->matched = 1;
        root->mask = mask;
        root->prefix = prefix;
    }
    if(leaf(root))
        return;
    if(!root->son0)
        root->son0 = btn_init(NULL, NULL,0);
    leaf_pushing(root->son0, root->prefix, root->mask);
    if(!root->son1)
        root->son1 = btn_init(NULL,NULL,0);
    leaf_pushing(root->son1, root->prefix, root->mask);
}
```

同样为递归构造，上面的节点将自己的值推下去，下面的节点判断一下是否比自己的 prefix 更长，如更长的话则更新，否则的话就保持原样。这样一个叶节点可能被多次更新，但仍然能保持最长前缀匹配的性质，只是可能需要的时间会稍微多些。

```
typedef struct new_tree_node{
    uint16_t bits;
    struct new_tree_node ** ina;    //array for intrenal node
    u32 * lna;    //array for leaf node prefix
} ntn;
```

开始构建新的多 bit 树，bits 取 16 位，满足最大要求的 4 位匹配。之后开始转换，由于转换部分代码较长，重复度较高，不再展示在报告中，详见 ip.c。

Match 部分与 btn 大同小异，直接放置在下面，不再详述：

```
//use fast trie to match ip
u32 fast_match(ntn * root, u32 ip, u32 mask, int start,int bit)
{
    u32 idx = (ip << (start - 1 )) >> (32 - bit);
    assert(root);
    if(root->bits & (1 << (15 - idx)))
    {
        assert(root->ina);
        ntn * new_root = root->ina[__builtin_popcount(root->bits >> (16 - idx))];
        return fast_match( new_root, ip, mask, start + bit, bit);
    }
    else
    {
        assert(root->lna);
        return root->lna[idx - __builtin_popcount(root->bits >> (16 - idx))];
    }
}
```

四、 实验结果

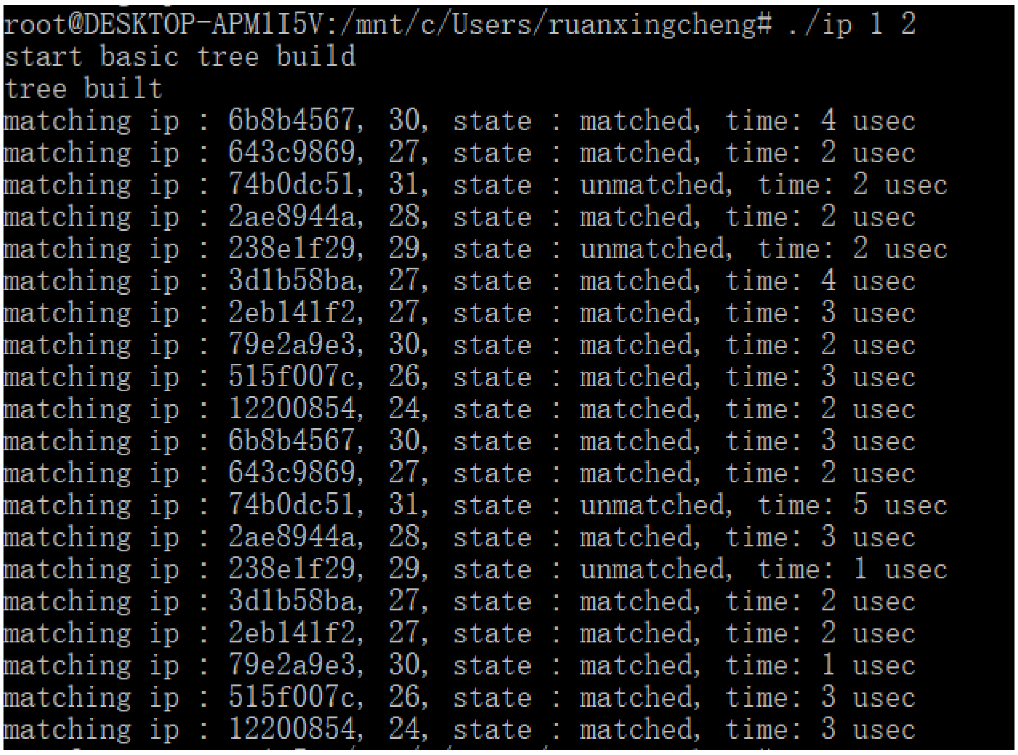
```
#define MATCH_TIMES 10000
```

通过修改最顶端的本项可以修改匹配的次数。

编译后运行有两个参数./ip a b，其中 a 表示匹配的方式，0 为基本的前缀匹配，1 为多 bit 前缀匹配，b 表示 bit 的数目，可选 1,2,3,4，请对应各自的前一个选项。

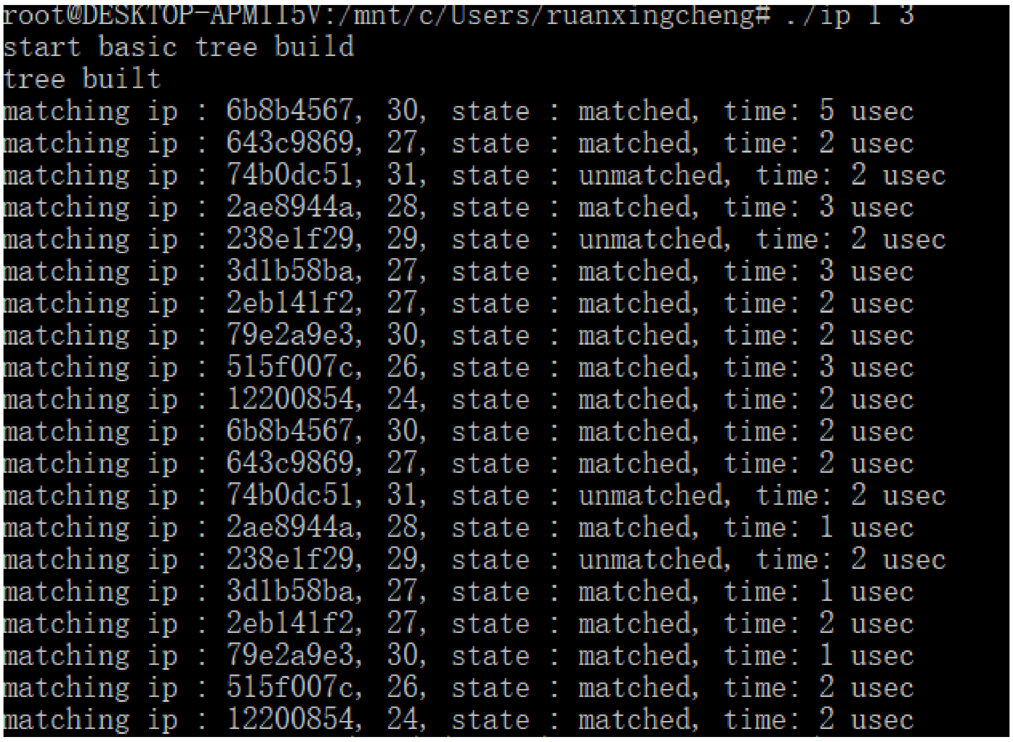
首先进行了单次的效果比较，单次的比对效果不明显，由于各种系统调用占据了大量的时间。

10 次比对，上基本，下 2bit



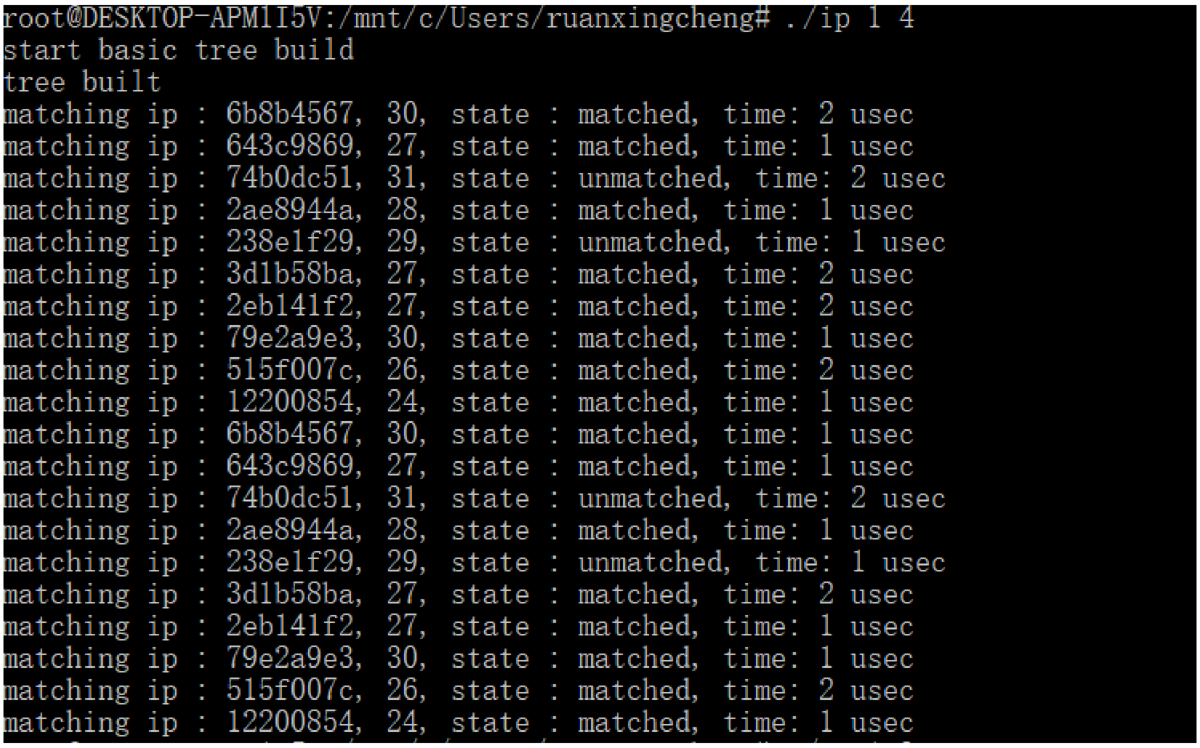
```
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
matching ip : 6b8b4567, 30, state : matched, time: 4 usec
matching ip : 643c9869, 27, state : matched, time: 2 usec
matching ip : 74b0dc51, 31, state : unmatched, time: 2 usec
matching ip : 2ae8944a, 28, state : matched, time: 2 usec
matching ip : 238e1f29, 29, state : unmatched, time: 2 usec
matching ip : 3d1b58ba, 27, state : matched, time: 4 usec
matching ip : 2eb141f2, 27, state : matched, time: 3 usec
matching ip : 79e2a9e3, 30, state : matched, time: 2 usec
matching ip : 515f007c, 26, state : matched, time: 3 usec
matching ip : 12200854, 24, state : matched, time: 2 usec
matching ip : 6b8b4567, 30, state : matched, time: 3 usec
matching ip : 643c9869, 27, state : matched, time: 2 usec
matching ip : 74b0dc51, 31, state : unmatched, time: 5 usec
matching ip : 2ae8944a, 28, state : matched, time: 3 usec
matching ip : 238e1f29, 29, state : unmatched, time: 1 usec
matching ip : 3d1b58ba, 27, state : matched, time: 2 usec
matching ip : 2eb141f2, 27, state : matched, time: 2 usec
matching ip : 79e2a9e3, 30, state : matched, time: 1 usec
matching ip : 515f007c, 26, state : matched, time: 3 usec
matching ip : 12200854, 24, state : matched, time: 3 usec
```

10 次比对，上基本，下 3 bit



```
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 3
start basic tree build
tree built
matching ip : 6b8b4567, 30, state : matched, time: 5 usec
matching ip : 643c9869, 27, state : matched, time: 2 usec
matching ip : 74b0dc51, 31, state : unmatched, time: 2 usec
matching ip : 2ae8944a, 28, state : matched, time: 3 usec
matching ip : 238e1f29, 29, state : unmatched, time: 2 usec
matching ip : 3d1b58ba, 27, state : matched, time: 3 usec
matching ip : 2eb141f2, 27, state : matched, time: 2 usec
matching ip : 79e2a9e3, 30, state : matched, time: 2 usec
matching ip : 515f007c, 26, state : matched, time: 3 usec
matching ip : 12200854, 24, state : matched, time: 2 usec
matching ip : 6b8b4567, 30, state : matched, time: 2 usec
matching ip : 643c9869, 27, state : matched, time: 2 usec
matching ip : 74b0dc51, 31, state : unmatched, time: 2 usec
matching ip : 2ae8944a, 28, state : matched, time: 1 usec
matching ip : 238e1f29, 29, state : unmatched, time: 2 usec
matching ip : 3d1b58ba, 27, state : matched, time: 1 usec
matching ip : 2eb141f2, 27, state : matched, time: 2 usec
matching ip : 79e2a9e3, 30, state : matched, time: 1 usec
matching ip : 515f007c, 26, state : matched, time: 2 usec
matching ip : 12200854, 24, state : matched, time: 2 usec
```

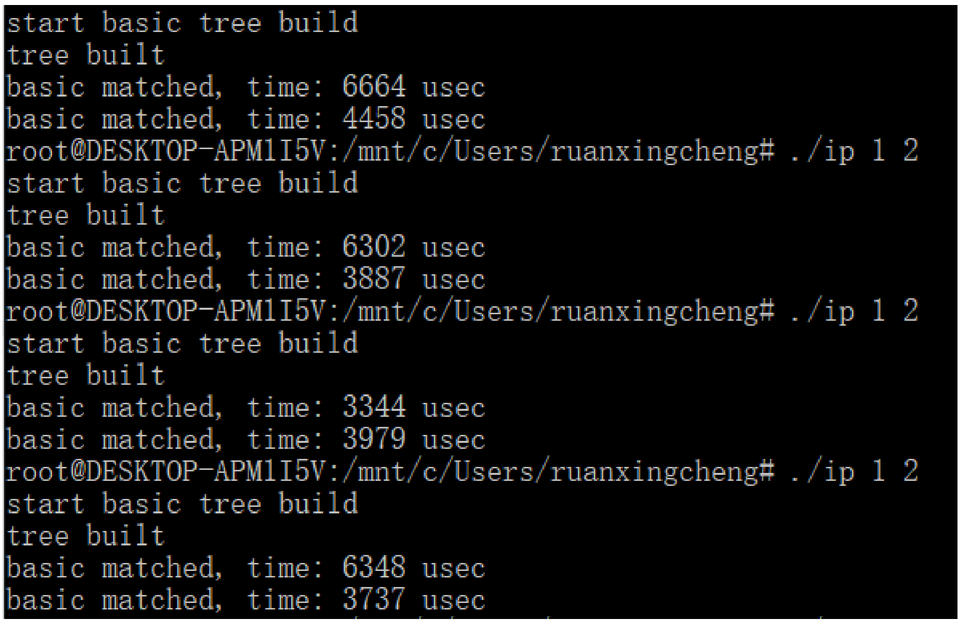
10 次比对，上基本，下 4bit



```
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 4
start basic tree build
tree built
matching ip : 6b8b4567, 30, state : matched, time: 2 usec
matching ip : 643c9869, 27, state : matched, time: 1 usec
matching ip : 74b0dc51, 31, state : unmatched, time: 2 usec
matching ip : 2ae8944a, 28, state : matched, time: 1 usec
matching ip : 238e1f29, 29, state : unmatched, time: 1 usec
matching ip : 3d1b58ba, 27, state : matched, time: 2 usec
matching ip : 2eb141f2, 27, state : matched, time: 2 usec
matching ip : 79e2a9e3, 30, state : matched, time: 1 usec
matching ip : 515f007c, 26, state : matched, time: 2 usec
matching ip : 12200854, 24, state : matched, time: 1 usec
matching ip : 6b8b4567, 30, state : matched, time: 1 usec
matching ip : 643c9869, 27, state : matched, time: 1 usec
matching ip : 74b0dc51, 31, state : unmatched, time: 2 usec
matching ip : 2ae8944a, 28, state : matched, time: 1 usec
matching ip : 238e1f29, 29, state : unmatched, time: 1 usec
matching ip : 3d1b58ba, 27, state : matched, time: 2 usec
matching ip : 2eb141f2, 27, state : matched, time: 1 usec
matching ip : 79e2a9e3, 30, state : matched, time: 1 usec
matching ip : 515f007c, 26, state : matched, time: 2 usec
matching ip : 12200854, 24, state : matched, time: 1 usec
```

偶然性较大，故进行了多次匹配的统计时间比较。

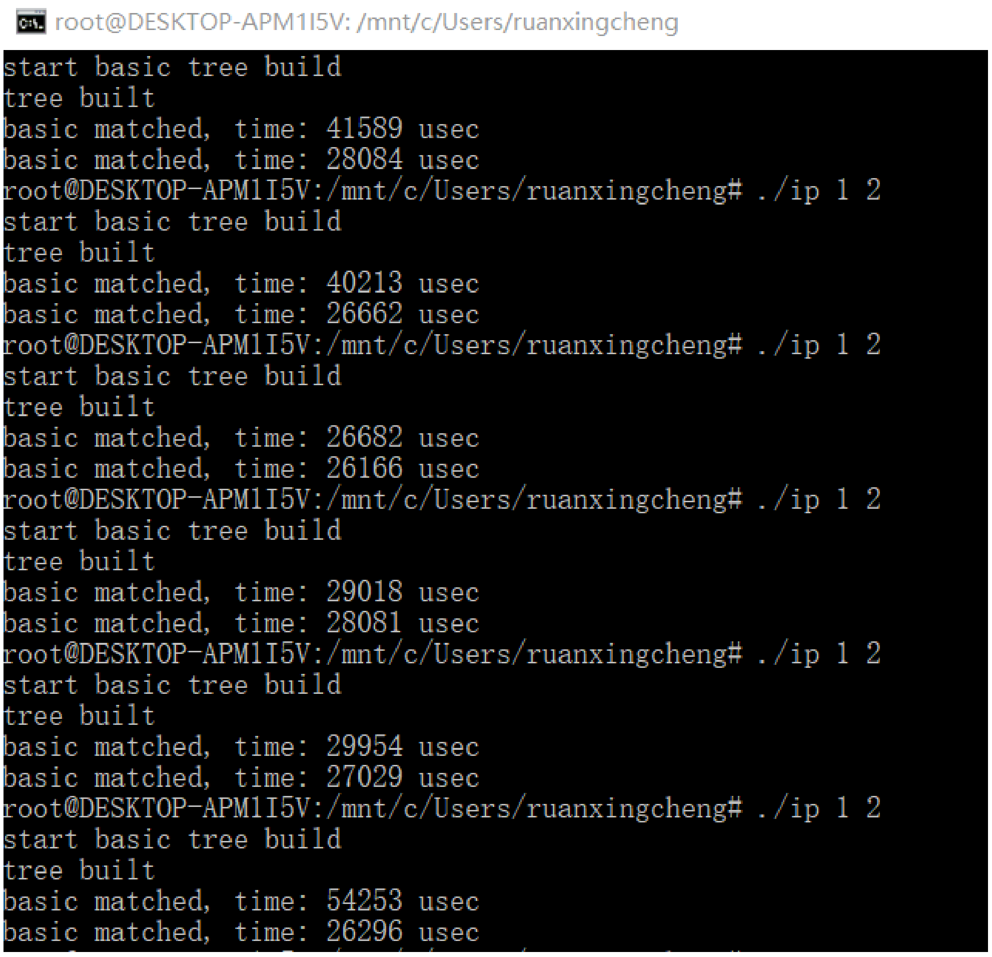
10,000 次比对，上基本，下 2bit



```
start basic tree build
tree built
basic matched, time: 6664 usec
basic matched, time: 4458 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 6302 usec
basic matched, time: 3887 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 3344 usec
basic matched, time: 3979 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 6348 usec
basic matched, time: 3737 usec
```

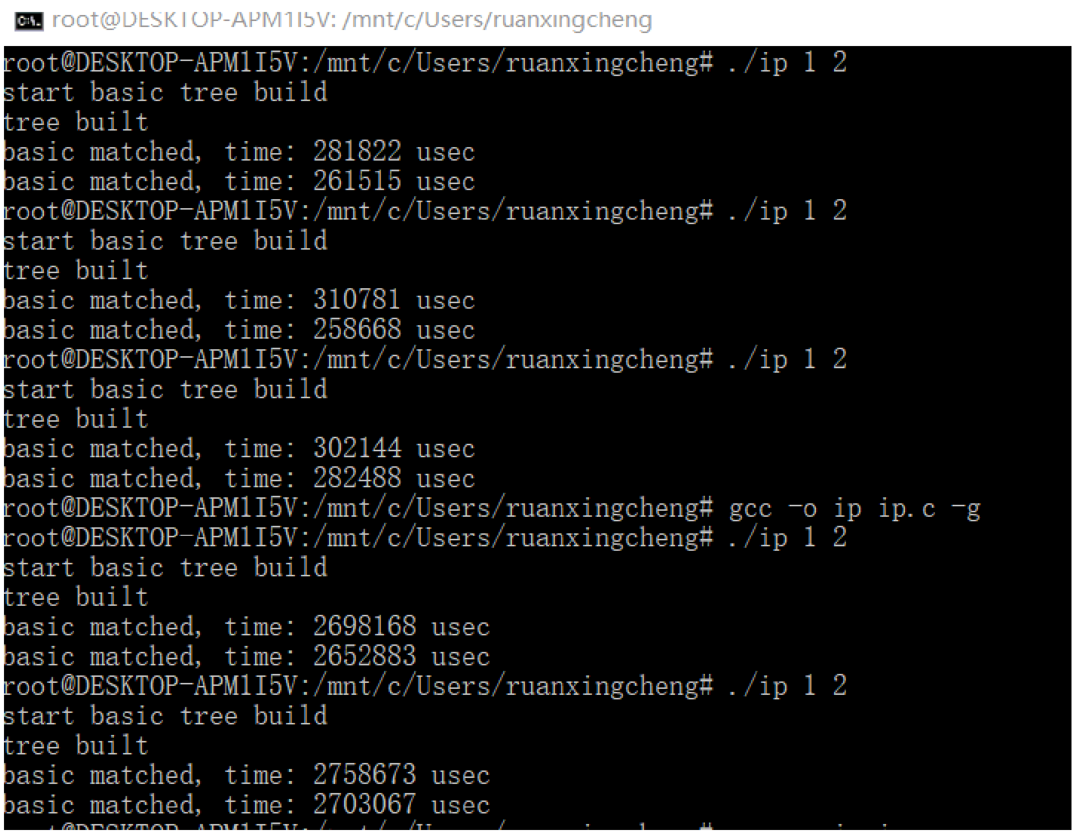
发现基本匹配的时间很不稳定，一些情况下较 2bit 来说要慢，但一些情况又会比 2bit 要快，于是换了次数再次测试

100,000 次比对，上基本，下 2bit



```
root@DESKTOP-APM1I5V: /mnt/c/Users/ruanxingcheng
start basic tree build
tree built
basic matched, time: 41589 usec
basic matched, time: 28084 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 40213 usec
basic matched, time: 26662 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 26682 usec
basic matched, time: 26166 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 29018 usec
basic matched, time: 28081 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 29954 usec
basic matched, time: 27029 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 54253 usec
basic matched, time: 26296 usec
```

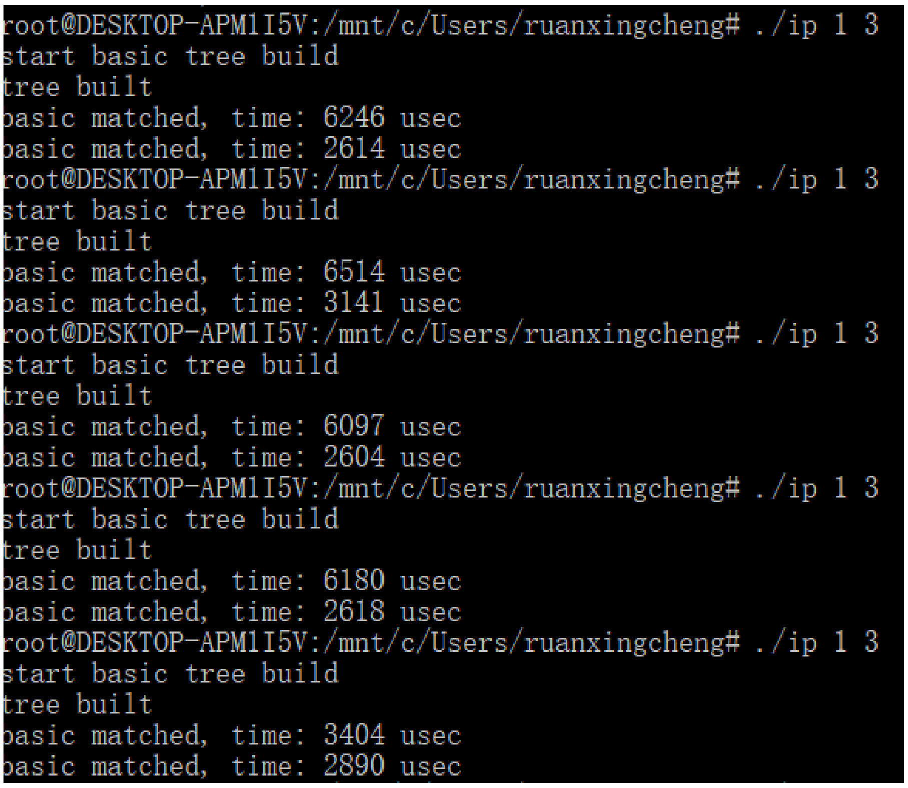
10,000,000 次比对，上基本，下 2bit


```
root@DESKTOP-APM1I5V: /mnt/c/Users/ruanxingcheng
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 281822 usec
basic matched, time: 261515 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 310781 usec
basic matched, time: 258668 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 302144 usec
basic matched, time: 282488 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# gcc -o ip ip.c -g
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 2698168 usec
basic matched, time: 2652883 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 2
start basic tree build
tree built
basic matched, time: 2758673 usec
basic matched, time: 2703067 usec
```

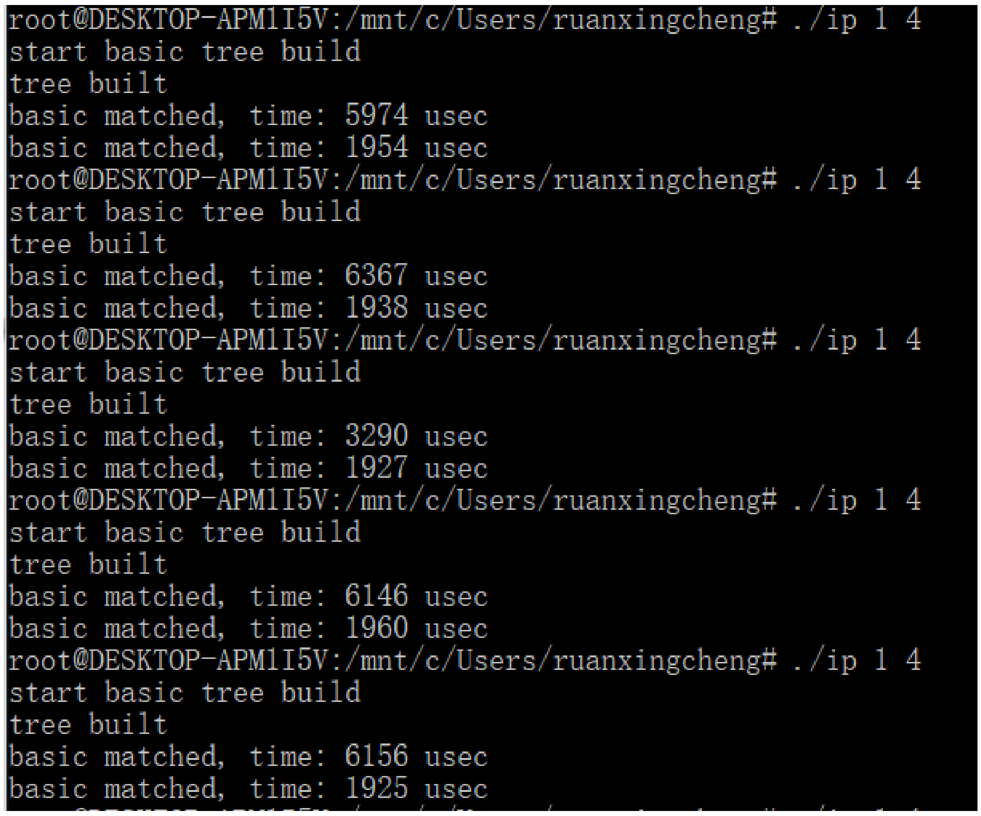
可见在 100,000 的时候这个趋势仍然保持着，而到了 10,000,000 次之后差别便不是那么明显了，还是可以看到基本的有波动，2bit 的时间比较稳定。波动的原因仍未找到。（匹配的序列由 rand 函数随机生成，两个匹配的序列一致，几次重复并不会改变 rand 函数的结果）

为展现区别，所以取了 10,000 为匹配次数，继续匹配剩下的 3bit 和 4bit。

10,000 次比对，上基本，下 3bit


```
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 3
start basic tree build
tree built
basic matched, time: 6246 usec
basic matched, time: 2614 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 3
start basic tree build
tree built
basic matched, time: 6514 usec
basic matched, time: 3141 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 3
start basic tree build
tree built
basic matched, time: 6097 usec
basic matched, time: 2604 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 3
start basic tree build
tree built
basic matched, time: 6180 usec
basic matched, time: 2618 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 3
start basic tree build
tree built
basic matched, time: 3404 usec
basic matched, time: 2890 usec
```

10,000 次比对，上基本，下 4bit


```
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 4
start basic tree build
tree built
basic matched, time: 5974 usec
basic matched, time: 1954 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 4
start basic tree build
tree built
basic matched, time: 6367 usec
basic matched, time: 1938 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 4
start basic tree build
tree built
basic matched, time: 3290 usec
basic matched, time: 1927 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 4
start basic tree build
tree built
basic matched, time: 6146 usec
basic matched, time: 1960 usec
root@DESKTOP-APM1I5V:/mnt/c/Users/ruanxingcheng# ./ip 1 4
start basic tree build
tree built
basic matched, time: 6156 usec
basic matched, time: 1925 usec
```

原来的趋势仍然存在，可以看到 3bit、4bit 相较原有的 2bit 有明显地提速，符合预期。

五、 实验总结

通过本次实现，我更加详细地了解了 IP 前缀查找的过程，以及明白了加速这个过程的一些手段，但是还有一些问题没有得到解决，我也还在思考当中。

本次实验书写中使用了很多递归，毕竟递归对于树结构来说是非常自然和必要的。递归的好处就在于代码更加简洁，实现更加方便，但同时也带来了调试的困难和更高的逻辑要求。整体来说实现下来比较轻松，但是调试的时候会由于一些刁钻的问题而摸不着头脑，比如说这次实验中的某个 else if 分支忘记写返回了，编译器在编译的时候用了离结束最近时的一个函数调用的返回充当了函数的返回值，导致出现了递归错误，2bit 层本应得到 1bit 层的结果却得到了 2bit 层的结果，匪夷所思，思考了很久。