Acwing · 秦淮岸灯火阑珊 2018.6.7 晚22:00

⑧ set 集合

- insert() 插入一个数
- find() 返回位置，无则 end
- count() 某一个数的个数
- erase()
  - 输入一个数是x，删除所有x $O(k+\log n)$
  - 输入一个迭代器，删除这个迭代器
- lower_bound() 返回大于等于x最小数迭代器
- upper_bound() 返回大于x最小的数的迭代器
- begin() 起点
- end() 终点
- ++；-- 返回前驱；后继

---

⑨ map 索引映射 平衡树

- insert() 加数
- find() 查找 pair类型
- [] 核心：数组使用 $O(\log n)$
- lower_bound / upper_bound 同上

⑩ unordered系列 同上，增删改查 $O(1)$

不支持 lower，upper，迭代器，无序

⑪ bitset 位运算 $\frac{1}{8}$内存，$\frac{1}{64}$时间，只有0和1.

- ~，&，|，^，位运算
- >>，<<，!，==，!=
- [] 任意位

count 返回多少位

any 判断是否至少有

none 判断是否全为0

set 把所有位置成1

set(k,v) 将第k位变成v

reset 把所有位置成0

filp 等价于~ 取反

filp(k) 将第k位取反

Acwing · 秦淮岸灯火阑珊

2018.6.7
晚 21:30

① vector 变长数组. 倍增思想
② string 字符串
③ queue 队列
④ priority_queue 优先队列, 堆
⑤ deque 双端队列
⑥ stack 栈
⑦ set, map, multiset, multimap 二叉平衡树 (红黑树)
⑧ unordered_set, unordered_map 哈希
⑨ bitset 压位, 位运算, 状态压缩.

常用STL

④ queue 队列
- push() 队尾插入
- pop() 队头弹出
- front() 队头
- back() 队尾
- empty() 判空
- size() 大小

⑤ priority_queue 优先队列, 堆
① 默认大根堆
- push() 压入堆
- pop() 弹堆顶

小根堆
- 存数时, 加负号 q.push(-x)
- priority_queue<int, vector<int>, greater<int>>

top() 堆顶

① vector 常数为2
vector<int> a(10, 3); 长度为10, 每位为3
a[10] 10个vector.

- a.size(); 多少位
- a.empty(); 是否空
- a.clear(); 清空

O(1)
- back() 尾部元素
- front() 头部元素
- push_back() 尾部插入
- pop_back() 头部插入
- begin() 起始 迭代器 a[0]
- end() 结尾 a[a.size()]

① 多开空间, 少开申请次数
② 支持比较运算 (按字典序)

---

② pair 二元组 可内装各种容器
- a.first 第一元素
- a.second 第二元素
- ① 支持比较元素
- ② 以first第一关键字 second第二关键字 字典序比较

make_pair(x, y); 二元组(x, y)

③ string 字符串
string a="yxc"; 赋值
a += "Acwing"; 连接
a += 'c';
a.substr(1, 2); 起始位置, 多少位

④ stack 栈
- push() 压数 (栈顶)
- pop() 弹数 (栈顶)
- top() 栈顶
- size() 长度
- empty() 空

⑦ deque 双端队列, 效率低
- push_back() 压入队尾
- push_front() 压入队头
- pop_back() 弹出队尾
- pop_front() 弹出队头
- size() 长度
- empty() 空
- clear() 清空
- begin() 头部
- end() 尾部
- [] 数组下标

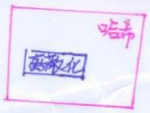
Acwing · 秦淮岸 · 灯火阑珊

2019
6.7
晚 20:00

哈希表

哈希表 — 存储结构 — 开放寻址 / 拉链法 (解决冲突)
哈希表 — 字符串 Hash

与离散化关系



离散化 ∈ 哈希
即一种有序的特殊哈希

将一个庞大的值域，映射到 [0, n] 一个较小的集合

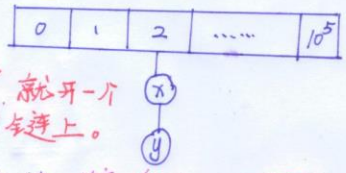
$0 \sim 10^9 \longrightarrow 0 \sim 10^5$ 常用情境

Acwing 840 模拟散列表 操作少，值域大。

哈希函数
① 直接取模。(模数一定是质数，与 2^x 尽 如：$x \% 10^5$ $\in (0, 10^5)$ 最远，x 任意自然数)
② 冲突：两个不一样的数，映射成同一个数

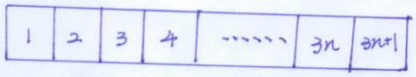
① 添加
② 查询
} 常用
③ 删除。一般开布尔数组标记
懒情删除法

拉链法



① 遇到冲突，就开一个结点，挂到链上。

开放寻址法 长度一般为 2~3 倍原长



$h(x) = k$
若 k 所在位已有数占领了，那么走到下一位。直到无人占领。

故 拉链法 竖向处理冲突
开放取址法 横向处理冲突

```
void insert (int x) 插入操作
{
    int k = (x%N + N) % N;
    +N. 是为了余数为正数
    edge[idx] = x;
    Next[idx] = head[k];
    head[k] = idx++;
}
void find (int x)
{
    int k = (x%N + N) % N; 哈希
    for (int i = head[k]; ~i; i = Next[i])
        if (edge[i] == x) 遍历链表
            return true; 找到了
    return false;
}
```

```
int find (int x)
{
    int k = (x%N + N) % N;
    while (h[k] != null && h[k] != x)
    {
        k++; 下一个
        if (k == N) 循环
            k = 0;
    }
    return k;
}
```

坑被占领　不是目标值

字符串 Hash

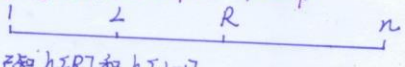
① 对于一个字符串，将它看成一个 P 进制数

$ABCD = (1234)_P = (1 \times P^3 + 2 \times P^2 + 3 \times P^1 + 4 \times P^0)$

① 一般情况，不能映射 0　mod Q 的方法出
A = 0
AA = 0 ⇒ 冲突

② 若无冲突，P = 131 或 13331，Q = 2^{64}　99.99% 不冲突

③ 利用前缀哈希，求子段哈希



已知 h[R] 和 h[L-1]

$P^{R-L+1} \times [P_0, P_{L-2}] = [P_0, P_{R-1}]$ 最重要的公式

$\Rightarrow h[R] - h[L] \times P^{R-L+1}$

秦淮岸·灯火阑珊 · Acwing

2019.6.8
搜索与图论
晚 20:05

树型

① 深度优先搜索 DFS. 执着选择一条路，就会一直往前走，直到无路可走，才会回溯.
数据结构：栈
空间：$O(h)$ 优于
不具最短路性质 劣于
最快深入
递归形式

图型 稳重

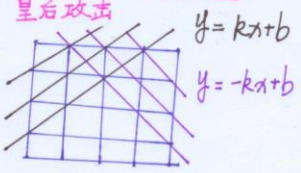
② 广度优先搜索 BFS. 前后八方，眼有六路
一层一层往外拓展，一层搜完后才往外走进入下一层.
数据结构：队列
空间：$O(2^h)$
最短路性质 即层次单调性质或单调居案性质
最少步骤
非递归形式

Acwing 八皇后

① 放一个皇后，需判 同一排，同一行，同一对角线 是否也有皇后. 可行性剪枝

① r[i] 判第i行
② c[i] 判第i列
③ dg[i+u] 判正对角线
④ udg[n-u+i] 判反对角线

皇后攻击



若图上每一条边的权都是一样的，则最短路就是图的层次 —— y总c

拓扑序（有向无环图）

① 每次将入度为0边的点，加入队列.
② 将入度为0的点，加入拓扑序，然后与之相连的点入度减一
③ 依此类推，直到所有点加入拓扑序

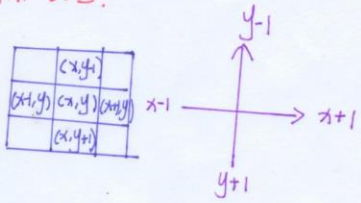
入度：b→a，则a入度++
出度：b→a，则b出度++
判环：拓扑序不完整

Acwing 走迷宫

凡是迷宫型问题，均可用 BFS.

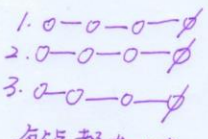
方向指示数组.

$dx[4] = \{1,-1,0,0\}$
$dy[4] = \{0,0,1,-1\}$



树 —— 无环连通图

存储方式
- 邻接矩阵 g[a,b] a→b 稠密图
- 邻接表 稀疏图



每个点都有一个表存连接的边

树的重心

① 选择一个点，将其作为根
② 若删此点，那么立所有的有联系的点各自成为连通块
③ 然后最大连通块，就是这个点的价值(代价)
④ 重心即价值最小的点.

size[x] 以x为根的子树大小，节点数目
n-size[x] 为删去x子树的剩余点，也是一个连通块。

Acwing · 秦淮岸 灯火阑珊

最短路 2018.6.14

最短路
建图
定义点和边

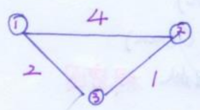
- 单源最短路
  - 所有边权都是正数
    - 朴素 Dijskra $O(n^2)$
    - 堆优化 Dijskra $O(m\log n)$
    - 线段树优化 Dijskra（无需看）
  - 存在负权边
    - Bellman-Ford $O(nm)$
    - SPFA 一般 $O(m)$，最坏 $O(nm)$
- 多源汇最短路 — Floyd $O(n^3)$

① 朴素 Dijskra

① $dist[s]=0$. / $dist[i]=INF$ } 初始化距离，S为起点、

② t 没有被访问的点、用 t 更新到其他点的距离

距离 dist[i] —— 从起点到 i 的最小距离



1 → 2 原本为4，但 1→3、3→2 的最短距离为3，故更新正确

② 堆优化

① 开一个堆，找权值最小的点，即 dist 最小

③ Bell-man 不能有负权回路 $O(nm)$

```
for (i=1; i<=n; i++)          // S号点经过不超过n条路的最短距离
    for (j=1; j<=m; j++)      (a,b,w)
        dist[b] = min(dist[b], dist[a]+w);
```

(a,b,w) 边定义　三角不等式 => 松弛操作

④ SPFA 队列优化

Acwing · 秦淮岸灯火阑珊

2019.6.15

算法基础树

搜索与图论（三）

最小生成树
- 普利姆
  - 朴素prim $O(n^2)$ 稠密图
  - 堆优化prim $O(m\log n)$ 不常用
- 克鲁斯卡尔 Kruskal $O(m\log m)$ 等价于 $O(m\log n)$ 稀疏图

二分图
- 判别：染色法 $O(n+m)$ 基于DFS
- 匈牙利
  - 最坏 $O(mn)$
  - 实际、远小于 $O(mn)$

① 朴素prim 与Dijkstra极为相似 稠密图

1' 初始化

2' n次迭代
- 找不在集合的点（最近）
- 集合一在连通块的最小点、
- 将上面的点更新集合的距离

st[t]=true 此点已加入集合

② kruskal算法

1' 将所有边 按照权从小到大排序 $O(m\log m)$

2' 枚举每条边 a—c—b 即(a,b,c)
- 若a,b不连通，则合并a,b，同时 ans+=c 并查集维护
- 若a,b连通，则 直接枚举下一边

③ 二分图 环中边的个数

① 当且仅当 不含有奇数 环

② 点集内部 没有边

③ 一个点染色 则所有点颜色都确定

即染色过程一定不矛盾

④ 匈牙利算法 二分图求最大匹配

无两边共有一点、即成功匹配

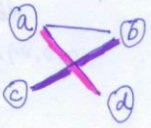
月老匹配问题

简化理解

每次匹配，若另一点已匹配，那试着给另一点原配，换一个匹配。

a love b
c love b
a love d

⇒ a love d
c love b



1' 染色法

```
for (int i=1; i<=n; i++)
  if (!color[i]) 未染色
  {
    if (!dfs(i,1)) 染色失败
    {
      flag = false;
      break;
    }
  }
```

```
bool dfs(int u, int c)
{
  color[u] = c;
  for (int i=head[u]; i; i=Next[i]) 遍历
  {
    int j=edge[i]; 取点
    if (!color[j]) { 未染色
      if (!dfs(j, 3-c)) 若c=1，则为2；若c=2，则为1
        return false;
    } else if (color[j]==c) 正好取反色
      return false; 已染，即两点一样，即 a=c, b=c 又(a,b)相连
  }
  return true;
}
```

Acwing 数学

欧拉函数 $\varphi(n)$ —— 1~n中与n互质数的个数

$\varphi(6) = 2$ （1, 5） 2个

$N = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_k^{a_k}$

$1 - \frac{1}{p_1} = \frac{p_1 - 1}{p_1}$

$1 - \frac{1}{p_2} = \frac{p_2 - 1}{p_2}$

$\varphi(n) = N \cdot (1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k})$ 容斥原理

$N = 6 = 2 \times 3$

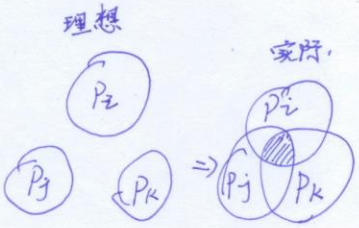
$\varphi(6) = 6 \times (1 - \frac{1}{2})(1 - \frac{1}{3}) = 6 \times \frac{1}{2} \times \frac{2}{3} = 2$

① 从1~N中去掉 $p_1, p_2, \cdots, p_k$ 的所有倍数

$N - \frac{N}{p_1} - \frac{N}{p_2} - \cdots - \frac{N}{p_k}$

② 加上 $p_i \times p_j$ 的倍数

$\Rightarrow + \frac{N}{p_1 p_2} + \frac{N}{p_1 p_3} + \cdots$ 两两组合



③ 减去 $p_i \times p_j \times p_k$ 的倍数

$\Rightarrow - \frac{N}{p_1 p_2 p_3} - \frac{N}{p_1 p_2 p_4} - \cdots$

④ 根据上面以此类推.

- - - - - - - - - - - - - -

当 i % phi[j] = 0 时

欧拉定理

a与n互质 $\Rightarrow a^{\varphi(n)} \equiv 1 \pmod n$

1~n中与a互质数有 $a_1, a_2, \cdots, a_{\varphi(n)}$

$a \cdot a_1, a \cdot a_2, \cdots a \cdot a_{\varphi(n)}$

习题课·秦淮岸·灯火阑珊

2018.6.5

基本数据结构

Acwing 143 最大异或对

```
  1 1 0 1 1 0
xor 0 0 1 1 0 1
-------------
  1 1 1 0 1 1
```

不进位加法.

亦 { a=b => 0 ; a!=b => 1 } 优化

a.b 均为 1 或 0.

= 进制处理

相同为 0

不同为 1

① 暴力算法 $O(n^2)$

两重循环枚举.

1' 枚举第 1 个数

2' 枚举第 2 个数 实质找和 a 异或值最大的数

```
ans = max(ans, a^b);
```

注: ^ 是 C++ 异或 (xor) 的符号

② Trie. (优化内层循环)

① 为什么使用 Trie?

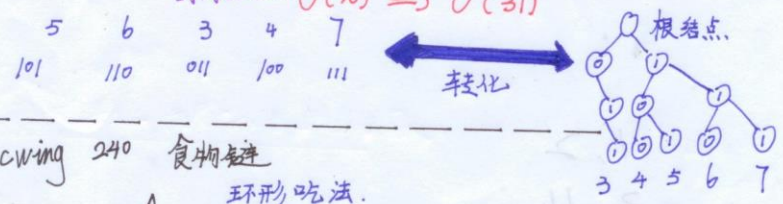
因为每个数可以看成 32 位二进数. 可用字符串存储

② 如何找最优匹配. 使异值最大?

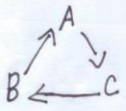
1' 每次找和当前位值相反或的数 (找不到就只能找相同的树分支)

简单来说. 就是走"唱反调"的分支。

时间优化: $O(n) \Rightarrow O(31)$



Acwing 240 食物链



环形吃法.

草克水
水克火
火克草

简单理解

具体建议看原题

假设: ① 冲突

② 不在范围内

③ 与关系冲突 (捕食)

一个动物有三种关系和其他动物

① 同类

② 捕食

③ 天敌

对于结点 x 与根结点关系

距离余 ① : 可以吃根节点.

② : 被根节点吃

③ : 和根节点是同类