Graph

Zty

ZUCC ACM Group

2023.01.06

目录

目录

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关
- 一条边连接两个节点

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关
- 一条边连接两个节点
- 权值：边的“费用”，可以形象地理解为边的长度。

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关
- 一条边连接两个节点
- 权值：边的“费用”，可以形象地理解为边的长度。
- 连通：如果图中结点U，V之间存在一条从U通过若干条边、点到达V的通路，则称U、V 是连通的。

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关
- 一条边连接两个节点
- 权值：边的“费用”，可以形象地理解为边的长度。
- 连通：如果图中结点U，V之间存在一条从U通过若干条边、点到达V的通路，则称U、V 是连通的。
- 回路：起点和终点相同的路径，称为回路，或“环”。

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关
- 一条边连接两个节点
- 权值：边的“费用”，可以形象地理解为边的长度。
- 连通：如果图中结点U，V之间存在一条从U通过若干条边、点到达V的通路，则称U、V 是连通的。
- 回路：起点和终点相同的路径，称为回路，或“环”。
- 完全图：一个n 阶的完全无向图含有n*(n-1)/2 条边；一个n 阶的完全有向图含有n*(n-1)条边；

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关
- 一条边连接两个节点
- 权值：边的“费用”，可以形象地理解为边的长度。
- 连通：如果图中结点U，V之间存在一条从U通过若干条边、点到达V的通路，则称U、V 是连通的。
- 回路：起点和终点相同的路径，称为回路，或“环”。
- 完全图：一个n 阶的完全无向图含有n*(n-1)/2 条边；一个n 阶的完全有向图含有n*(n-1)条边；
- 稠密图：一个边数接近完全图的图。
  稀疏图：一个边数远远少于完全图的图。。

图的基本概念

- 图由点和边组成，是n个结点 m条边构成，n和m无关
- 一条边连接两个节点
- 权值：边的“费用”，可以形象地理解为边的长度。
- 连通：如果图中结点U，V之间存在一条从U通过若干条边、点到达V的通路，则称U、V 是连通的。
- 回路：起点和终点相同的路径，称为回路，或“环”。
- 完全图：一个n 阶的完全无向图含有n*(n-1)/2 条边；一个n 阶的完全有向图含有n*(n-1)条边；
- 稠密图：一个边数接近完全图的图。
  稀疏图：一个边数远远少于完全图的图。。
- 简单路径：路径上，每个顶点都不相同（即，没有环）

图的基本概念

- 无向图的边是双向的,u能到v,v一定能到u

图的基本概念

- 无向图的边是双向的,u能到v,v一定能到u
- 结点的度：无向图中与结点相连的边的数目，称为结点的度。

图的基本概念

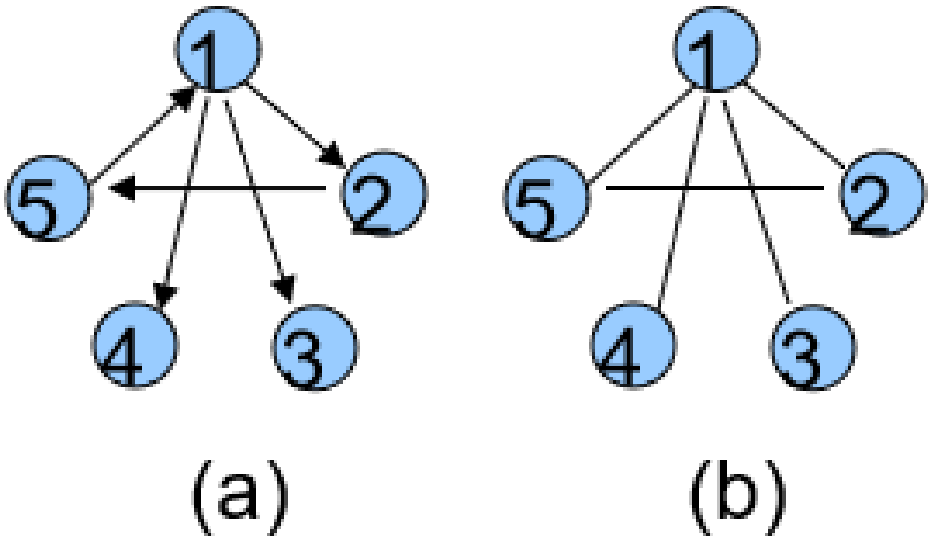
- 无向图的边是双向的,u能到v,v一定能到u
- 结点的度：无向图中与结点相连的边的数目，称为结点的度。
- 有向图的边是单向的,u能到v,v不一定能到u

图的基本概念

- 无向图的边是双向的,u能到v,v一定能到u
- 结点的度：无向图中与结点相连的边的数目，称为结点的度。
- 有向图的边是单向的,u能到v,v不一定能到u
- 结点的入度：在有向图中，以这个结点为终点的有向边的数目。
  结点的出度：在有向图中，以这个结点为起点的有向边的数目。

图的基本概念-图示

例如下图, a是含有5个节点, 5条边的有向图
b是含有5个节点, 5条边的无向图
a中1的出度为3, 入度为1
b中1的度为4
a中2到3是连通的, 但3到2是不连通的
b中2到3是连通的, 3到2也是连通的



图的存储

记录每个结点直连的点

- 每个结点会有多个直连的点，可以采用二维数组

  ```
  int e[114514][114514];
  ```

图的存储

记录每个结点直连的点

- 每个结点会有多个直连的点，可以采用二维数组

  int e[114514][114514];
- 用e[u][v]来表示结点u和v之间有一条直连的边

图的存储

记录每个结点直连的点

- 每个结点会有多个直连的点，可以采用二维数组

  int e[114514][114514];

- 用e[u][v]来表示结点u和v之间有一条直连的边
- 显然直接使用二维数组会MLE，于是可以采用二维数组 + 动态数组的方法

  vector<int> e[114514];

图的存储

记录每个结点直连的点

- 每个结点会有多个直连的点, 可以采用二维数组

  int e[114514][114514];

- 用e[u][v]来表示结点u和v之间有一条直连的边
- 显然直接使用二维数组会MLE, 于是可以采用二维数组 + 动态数组的方法

  vector<int> e[114514];

- 这样空间就从$n \times n$ 被优化到了 $\sum e[i].size() = m$, 避免了MLE

目录

bfs

- 从一个节点开始，访问他所有直连的点，然后访问所有直连点的点

```cpp
vector<int> e[N];
int vis[N];
void bfs(int x) {
    queue<int> q;
    q.push(x);
    vis[x] = 1;
    while (!q.empty()) {
        int now = q.front();
        for (auto u : e[now]) {
            if (vis[u]) continue;
            vis[u] = 1;
            q.push(now);
        }
    }
}
```

dfs

- 从一个节点开始，访问某个直连的点，然后访问直连点的某个直连点

```
1: function DFS(u)
2:     if u has been visited then
3:         return
4:     end if
5:     visit u
6:     for v ← edge of u do
7:         DFS(v)
8:     end for
9: end function
```

目录

拓扑

- 一个无环的有向图称作有向无环图，Directed acyclic graph，简称DAG

拓扑

- 一个无环的有向图称作有向无环图，Directed acyclic graph，简称DAG
- 可以跑一遍拓扑来判断一张图有没有环

拓扑

- 一个无环的有向图称作有向无环图，Directed acyclic graph，简称DAG
- 可以跑一遍拓扑来判断一张图有没有环
- 在DAG上跑拓扑能得到拓扑序

拓扑实现

- 其实就是一种特殊的bfs

```cpp
vector<int> e[N];
int du[N];
void bfs(int x) {
    queue<int> q;
    for (int i = 1; i <= n; ++i) {
        if (du[i] == 0) q.push(i);
    }
    while (!q.empty()) {
        int now = q.front();
        for (auto u : e[now]) {
            du[u]--;
            if (du[u] == 0) q.push(now);
        }
    }
}
```

拓扑-例题

旅行计划

Mouse Hunt

目录

最短路

求在一张图中u到v的最短路径长度

- 多源最短路，跑一次得出n个点两两之间的最短路径长度
- 单源最短路，跑一次得出某个点到其他n-1个点之间的最短路径长度

多源最短路(Floyd)

原理

- 复杂度(n^3)
- 用邻接矩阵存，记得初值要赋成inf
- 实现是用三层for循环，枚举中间节点k，再枚举u，最后枚举v
  dis[u][v] = min(dis[u][v],dis[u][k] + dis[k][v])

多源最短路(Floyd)

例题

Anna, Svyatoslav and Maps
String Problem

单源最短路(Dijkstra)

原理

- Dijkstra是一种贪心的最短路算法
- 它先求出长度最短的一条路径，再参照该最短路径求出长度次短的一条路径直到求出从源点到其他各个顶点的最短路径。
- 用Dijkstra求最短路要求边权非负

单源最短路(Dijkstra)

朴素的Dijkstra

- 将源点的dis赋为0，其他点都赋为inf
- for n次，每次都先for n次寻找没有用过的最近的点
- 枚举这个最近的所有边，更新每个点的dis
- 复杂度 $O(m + n^2)$

单源最短路(Dijkstra)

优先队列优化

- 将源点的dis赋为0，其他点都赋为inf
- 从优先队列中找到最近的点
- 枚举这个最近的所有边，更新每个点的dis
- 复杂度 $O(mlogm)$

单源最短路(Dijkstra)

例题

【模板】单源最短路径（标准版）

END