Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

по «Алгоритмам и структурам данных»

Алгоритмы на графах

Выполнил:

Студент группы P3233

Нгуен Нгок Дык

Преподаватели:

Косяков М.С.

Санкт-Петербург

2022

Code: https://github.com/ndwannafly/ITMO_ALGO

Задача № **M. Цивилизация**

Task reformulation:

- Given a N x M matrix.
- Matrix consists of three types of cell:

  '.' which has weight = 1

  'W' which has weight = 2

  '#' is forbidden
- Find shortest path start from (x,y) to (z,t)
- Print the weight of the path
- Print the directions

Constraint:

- $1 \leq N \leq 1000$
- $1 \leq M \leq 1000$

Keyword: BFS, queue, dynamic programming

Solution:

- F[i][j] = shortest path from (x,y) to (i,j)
- DP formula:
  - F[i][j] = min(F[i][j], F[p][q] + a[i][j]) ; (p,q) is the cell that can move to (i,j)
- BFS from (x,y) , spread to all the nearby cells.
- Maintain a tracing array trace[i][j] = (p, q) to trace the previous cell.

Complexity:

- Complexity of BFS = |V| + |E|
- In this case, |E| = 4 * |V|
- Operations: O(n * m).
- Space: O(n * m)

Задача № **N. Свинки-копилки**

Task reformulation:

- Given N-element arrays. Let’s call it as N dominos
- Domino i falls if domino a[i] falls
- Find the minimum times we push the domino in order to let all of them down

Constraint:

- $N \leq 100$

Keyword: Graph, DFS

Solution:

- Make directed edge from a[i] to i
- The graph consists of several of components.
- Observation: Graph is directed but we can consider it as undirected graph and solve. Because each components need one hit to let all of them down (by hitting vertex which has indegree = 0).
- Make graph undirected and count the number of components by DFS
- Result = number of components.

Complexity:

- Operations: O(n).
- Spaces: O(n).

Задача № **О. Долой списывание!**

Task reformulation:

- Given a graph N vertices, M edges.
- Check whether a graph is 2-colorable.

Constraint:

- $1 \leq N \leq 100$
- $0 \leq M \leq N * (N-1)/2$

My comment:

- Recall discrete math in first year at ITMO.
- 2-colorable graph is also called as bipartite graph
- There are many interesting things about bipartite graph but it seems that this course doesn’t cover it. But bipartite graph and maximum flow are my favorite topics.

Keyword: Graph, DFS, BFS, 2-colorable, , bipartite graph

Solution:

- DFS start with vertex u
- Assign all adjacent vertices to the color different from u
- Continue ...
- If we meet any v that v is already assigned and color[u] = color[v] => Graph is not 2-colorable. (*)
- Graph is 2-colorable if (*) does not occur.

Complexity:

- Operations : O(|V| + |E|)
- Spaces : O(|V| + |E|)

Задача № **P. Авиаперелёты**

Task reformulation:

- Given N x N adjacent matrix
- A[i][j] = cost go directly from i to j
- Cost of path from any u to v = Min(A[i][j]) ; (i,j) is arrow passing through on the path from u to v.
- Find minimum cost C such that path from any vertex u to any vertex v has cost <= C
- In other word, C = max(shortest_path[u][v]); with any vertices u and v

Constraint:

- 1 ≤ N ≤ 1 000
- 0 ≤ a[i][j] ≤ 10^9

Keyword: Shortest Path, Floyd-Warshall, Strongly Connected graph, Binary Search

My comment:

- In a sparse graph (less edges) , Dijkstra is a good choice to find shortest path from s to any vertex u.
- But the given graph is adjacent matrix and it also requires find shorest path of any two vertices u , v. So forget about Dijkstra.
- Floyd-Warshall is effective algorithm to find shortest path of any two vertices in adjacent matrix. Complexity: O(n^3), spaces O(n^2).
- But 1 ≤ N ≤ 1 000 so we have to come up with another idea. Let's discuss it after Floyd subtask.

Subtask Floyd O(n^3) :

- F[u][v] = shortest_path from any u to v
- DP formula:
  - F[u][v] = max(F[u][k], F[k][v]); (u -> k -> v)

Solution O(N^2logn):

- Binary search the minimum cost C.
- With each C, build the new graph that has arrow between u and v if and only if a[u][v] <= C

- Check whether the graph is a **strongly connected.**
- Strongly connected is a graph that there exists path between any two vertices.
- How to check?
- Easy to prove that:
  - We can say graph G is **strongly connected** if:

1. DFS from V visits all vertices in graph G, then there exists a path from V to every other vertex in G
2. There exists a path from every other vertex in G to V.

- To check (1), just DFS from V and check whether we visit all the vertices
- To check(2), reverse the edge and check same as (1)
- If (1) and (2) are true then G is **strongly connected**
- Otherwise, **NOT**

Complexity:

- Operation**:** O(n^2log(10^9)). Complexity of DFS in adjacent matrix = O(|V^2|)
- Spacing**:** O(n^2)