|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Cu | Downtown | Will Village | Neptune | 29th |
| Cu | 1 | 1 | 1 | 0 | 1 |
| Downtown | 1 | 1 | 0 | 0 | 1 |
| Williams Village | 1 | 0 | 1 | 1 | 1 |
| Neptune | 0 | 0 | 1 | 1 | 0 |
| 29th Street | 1 | 1 | 1 | 0 | 1 |

Adjacency matrix

Graph (V,E)

Includes V vertices and E edges, each location is a vertex, each edge is a bike path between vertices. If 1 in adjacent matrix, edge in a graph

Making an adjacency matrix with a graph

Williams Village

29th Street Mall

Neptune

Downtown

CU

Undirected graph: edge between vertices (A, B) is the same as between (B, A)

Adjacency list

Array of vertices: Each item in the array is an object. One property of linked lists is edges.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CU | Downtown | Williams Village | Neptune | 29th Street |

|  |
| --- |
| CU |
| Downtown |
| Williams Village |
| 29th Street |

Directed Graph

CU->downtown. Imagine bike paths become one way, all the paths from before are now one way paths.

29th->CU

Downtown

Neptune

Williams Village

29th Street Mall

CU

What is the distance from CU to 29th Street mall?

Weighted Graph

Distance is the weight

The graph represents some reality. Use the graph to solve a problem

Travelling salesman problem

We have a salesman Bob, he has to visit 5 cities and wants to do it as cheaply as possible. He wants to know to cheapest path through the graph returning to the starting node, visiting each of the cities exactly one time. Let’s say we have 5 cities, we will call them A, B, C, D, E. The weight will be the cost of the flight. In this flight to/from all citys.

C

D

B

E

A

For this case we have 24 possible routes 12 unique costs. In an exhaustive search we check each one, possible for few nodes. For 10 or more nodes it will not be possible.

Breadth-first search, depth-first search

Some other examples of travelling salesman

->Zoning school busses.

->Pick up kids with as few resources as possible.

->Package delivery – Deliver packages as quickly as possible, using the least resources