Data Structure & Algorithms

Sunbeam Infotech

Nilesh Ghule

1. bfs & dfs
2. min spanning tree
3. single src shortest path
4. all pair shortest path
5. point to point shortest path

A* Search Algorithm

- Point to point approximate shortest path finder algorithm.
- This algorithm is used in artificial intelligence.
- Commonly used in games or maps to find shortest distance in faster way.
- It is modification of BFS.

Priority queue

- Put selected adjacent vertices on queue, based on some heuristic.
- A math function is calculated for vertices
  - f(v) = g(v) + h(v) → vertex with min f(v) is picked.
  - g(v) → cost of source to vertex v
  - *h(v)* → estimated cost of vertex v to destination.

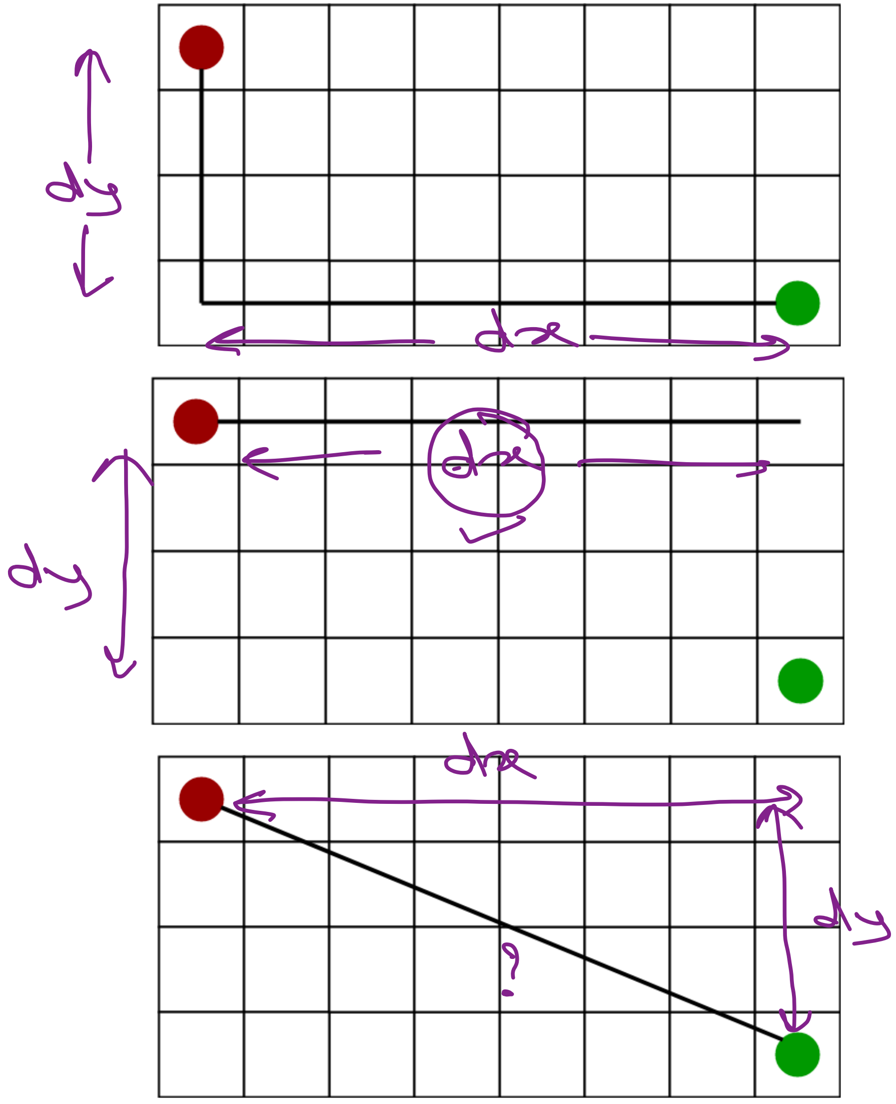
src

	0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	1	0	1	1
2	1	1	1	0	1	1	0	1	0	1
3	0	0	1	0	1	0	0	0	0	1
4	1	1	1	0	1	1	1	0	1	0
5	1	0	1	1	1	1	0	1	0	0
6	1	0	0	0	0	1	0	0	0	1
7	1	0	1	1	1	1	0 ?	1	1	1
8	1	1	1	0	0	0	1	0	0	1
9	1	0	1	0	0	0	0	0	0	1

dest

A* Search Algorithm

- h(v) represent heuristic and depends on problem domain. Three common techniques to calculate heuristic:
- Manhattan distance
  - When moves are limited in four directions only.
  - h = dx + dy
- Diagonal distance
  - When moves are allowed in all eight directions (one step).
  - h = MAX(dx, dy)
- Euclidean distance
  - When moves are allowed in any direction.
  - h = $\sqrt{dx^2 + dy^2}$
- Note that heuristic may result in longer paths in typical cases.



A* search algorithm

$f = g + h$

- Start point g(v) = 0, h(v) = 0 & f(h) = 0.
- Push start point vertex on a priority queue (by f(v)).
- Until queue is empty
  - Pop a point (v) from queue.
  - Add v into the path.
  - For each adjacent point (u)
    - If u is destination, build the path.
    - If u is invalid or already on path or blocked, skip it.
    - Calculate newg = g(v) + 1, newh = *heuristic* and newf = newg + newh.
    - If newf is less than f(u), f(u) = newf and also parent(u) = v. Rearrange elements in priority queue.

manhatten
diagonal
euclidean

	0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	1	0	1	1
2	1	1	1	0	1	1	0	1	0	1
3	0	0	1	0	1	0	0	0	0	1
4	1	1	1	0	1	1	1	0	1	0
5	1	0	1	1	1	1	0	1	0	0
6	1	0	0	0	0	1	0	0	0	1
7	1	0	1	1	1	1	0	1	1	1
8	1	1	1	0	0	0	1	0	0	1
9	1	0	1	0	0	0	0	0	0	1

A* Search Algorithm

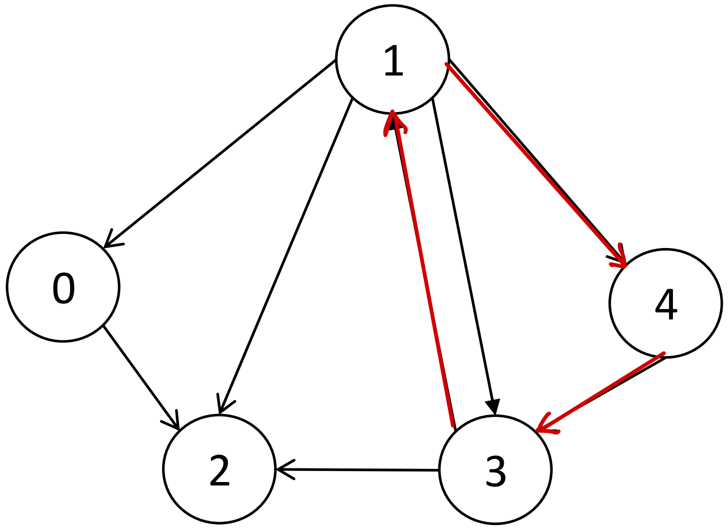
euclidean dist

	0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	1	0	1	1
2	1	1	1	0	1	1	0	1	0	1
3	0	0	1	0	1	0	0	0	0	1
4	1	1	1	0	1	1	1	0	1	0
5	1	0	1	1	1	1	0	1	0	0
6	1	0	0	0	0	1	0	0	0	1
7	1	0	1	1	1	1	0	1	1	1
8	1	1	1	0	0	0	1	0	0	1
9	1	0	1	0	0	0	0	0	0	1

	0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	1	0	1	1
2	1	1	1	0	1	1	0	1	0	1
3	0	0	1	0	1	0	0	0	0	1
4	1	1	1	0	1	1	1	0	1	0
5	1	0	1	1	1	1	0	1	0	0
6	1	0	0	0	0	1	0	0	0	1
7	1	0	1	1	1	1	0	1	1	1
8	1	1	1	0	0	0	1	0	0	1
9	1	0	1	0	0	0	0	0	0	1

Topological sort / order

- Topological order is linear ordering of graph vertices such that for every directed edge u-v, vertex u comes before vertex v.
- It is possible only for DAG.
- Directed Acyclic Graph
  - Edges have directions
  - Graph have no cycles
- Widely used in
  - Job scheduling → pig, spark, airflow, ...
  - Instructions re-ordering → RISC pipeline
  - Resolving symbols while linking → compiler design
  - Building components → Maven, Gradle, ...
  - Data serialization → serialization
  - Database backup & restore (FK)



1 0 4 3 2

topological sort

1 0

1 2

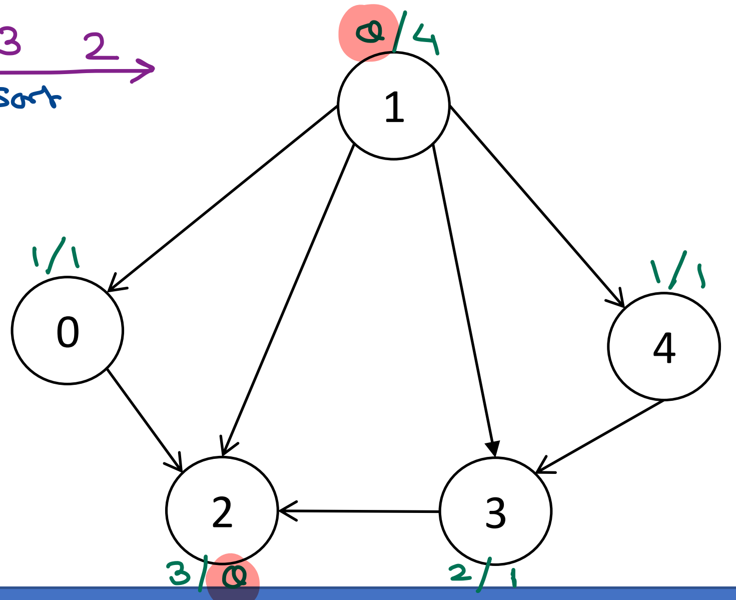
1 3

1 4

0 2

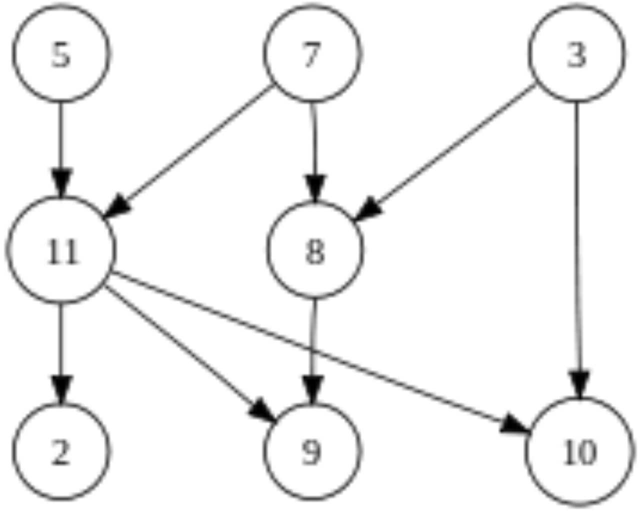
4 3

3 2

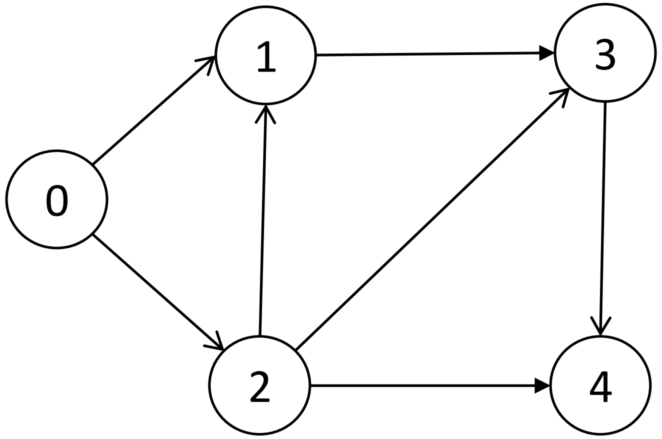


Topological order

- A graph may have multiple topological orders.
  - 5, 7, 3, 11, 8, 2, 9, 10 (visual left-to-right, top-to-bottom)
  - 3, 5, 7, 8, 11, 2, 9, 10 (smallest-numbered available vertex first)
  - 5, 7, 3, 8, 11, 10, 9, 2 (fewest edges first)
  - 7, 5, 11, 3, 10, 8, 9, 2 (largest-numbered available vertex first)
  - 5, 7, 11, 2, 3, 8, 9, 10 (attempting top-to-bottom, left-to-right)
  - 3, 7, 8, 5, 11, 10, 2, 9 (arbitrary)



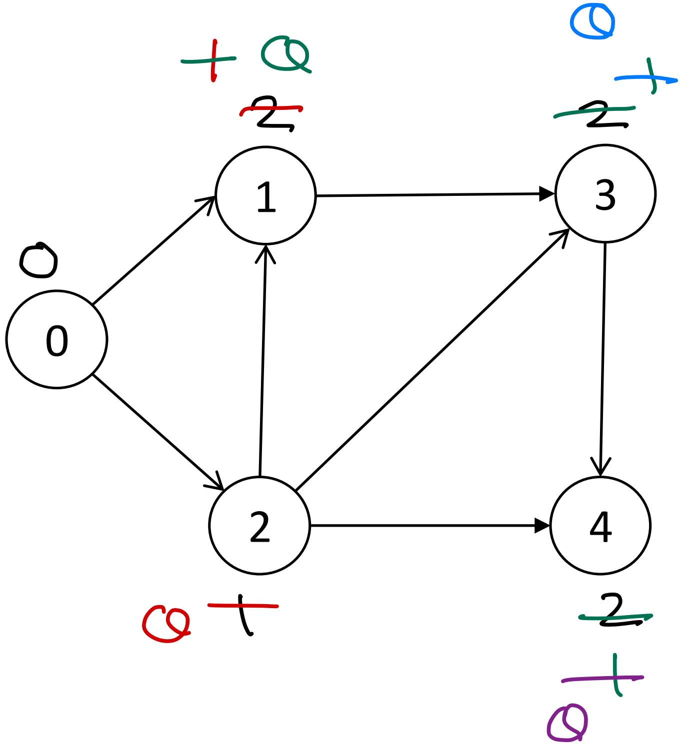
- Few graphs may have single topological order.
- DAG properties
  - Each DAG will have at least one vertex with out-degree 0.
  - Each DAG will have at least one vertex with in-degree 0.
- Topological order algorithms
  - DFS based algorithm
  - Kahn's algorithm



Topological order

- Kahn's algorithm
  - Calculate in-degree of all vertices in the graph.
  - Push all vertices with in-degree 0 on a queue.
  - Until queue is empty
    - Pop a vertex from the queue.
    - Add it into topological order
    - Decrement in-degree of each adjacent vertex.
    - If adjacent vertex in-degree become 0, push it on the queue.
  - If number of vertices in topological order is not same as number of vertices, topological order is not possible.

	0	1	2	3	4
0	x	1	1	x	x
1	x	x	x	1	x
2	x	1	x	1	1
3	x	x	x	x	1
4	x	x	x	x	x



0 2 1 3 4

Graph applications

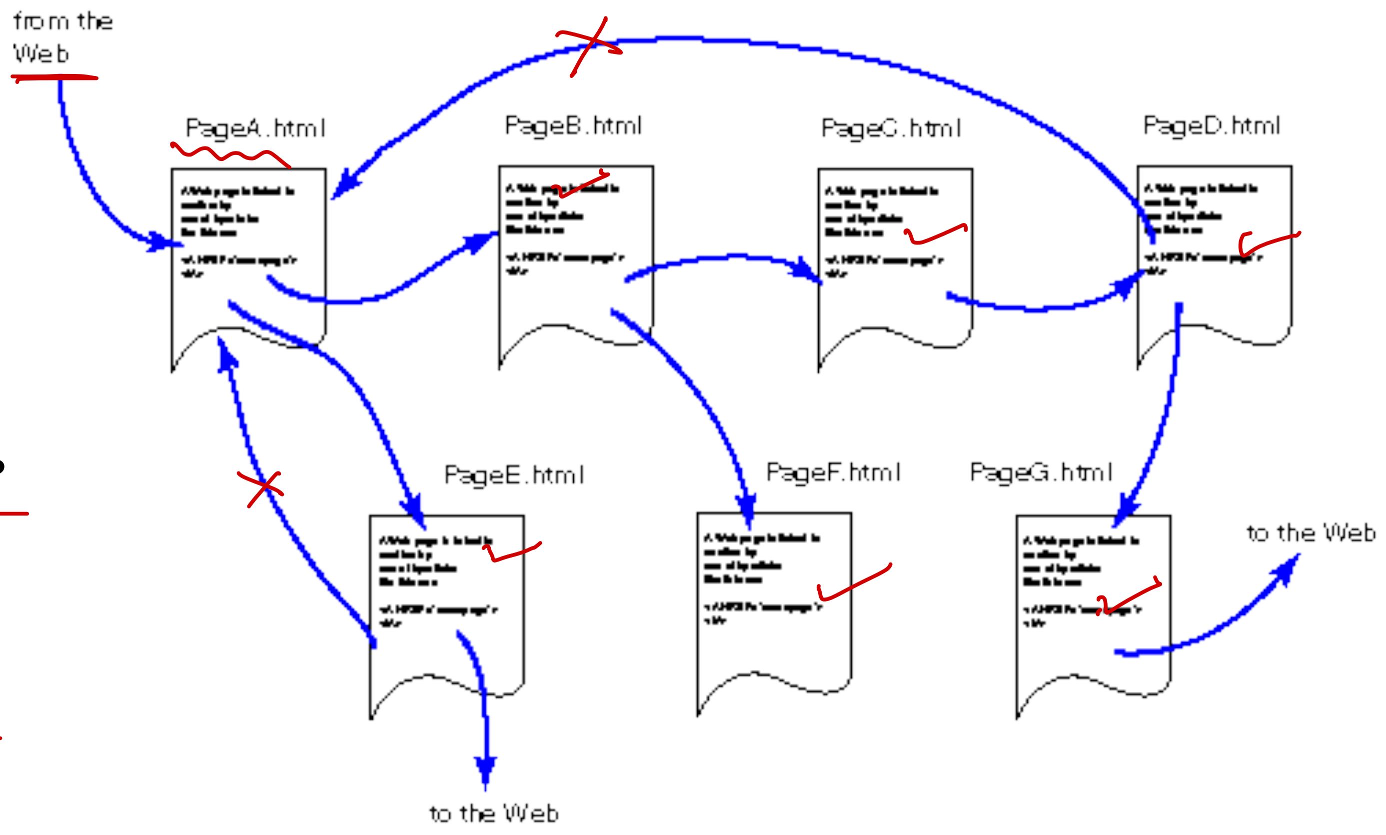
- Graph represents flow of computation/tasks. It is used for resource planning and scheduling. MST algorithms are used for resource conservation. DAG are used for scheduling in Spark or Tez.

- In OS, process and resources are treated as vertices and their usage is treated as edges. This resource allocation algorithm is used to detect deadlock.

- In social networking sites, each person is a vertex and their connection is an edge. In Facebook person search or friend suggestion algorithms use graph concepts.

Graph applications

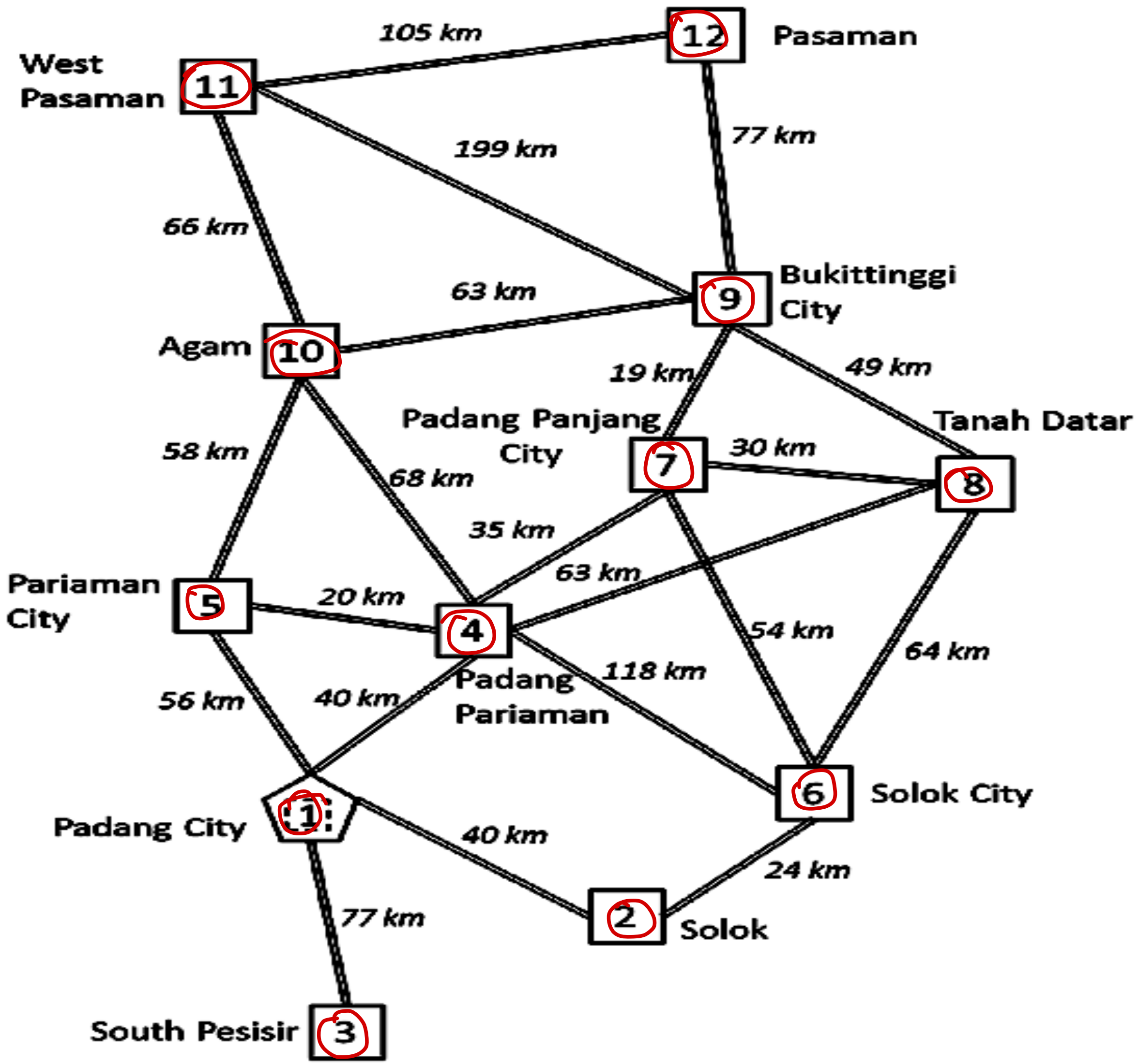
- In world wide web, web pages are like vertices; while links represents edges. This concept can be used at multiple places.
  - Making sitemap
  - Downloading website or resources
  - Developing web crawlers
  - Google page-rank algorithm



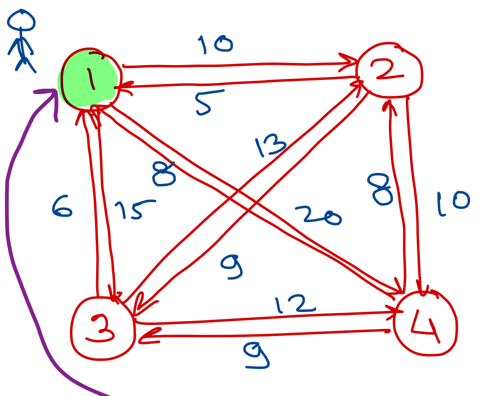
PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

Graph applications

- Maps uses graphs for showing routes and finding shortest paths. Intersection of two (or more) roads is considered as vertex and the road connecting two vertices is considered to be an edge.



travelling salesman problem DP ✓ brute-force

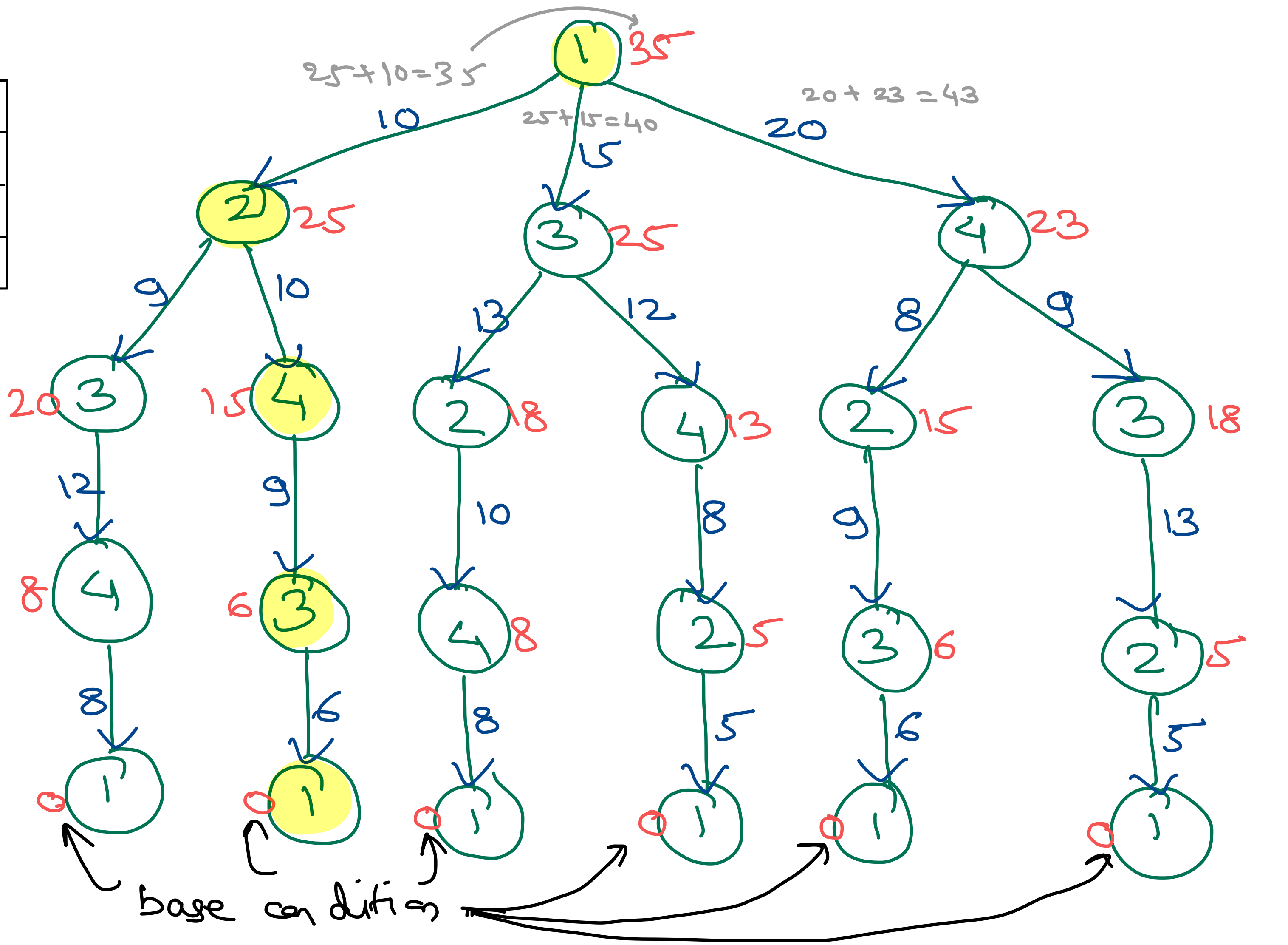


	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$$dist(s, V-s) = \min\left(wt(s,k) + dist(k, V-k) \right)$$

recurrence relation

S → start vertex.
V → set of all vertices
V-s → set of all vertices except s.
V-k → set of all vertices except k.



Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>