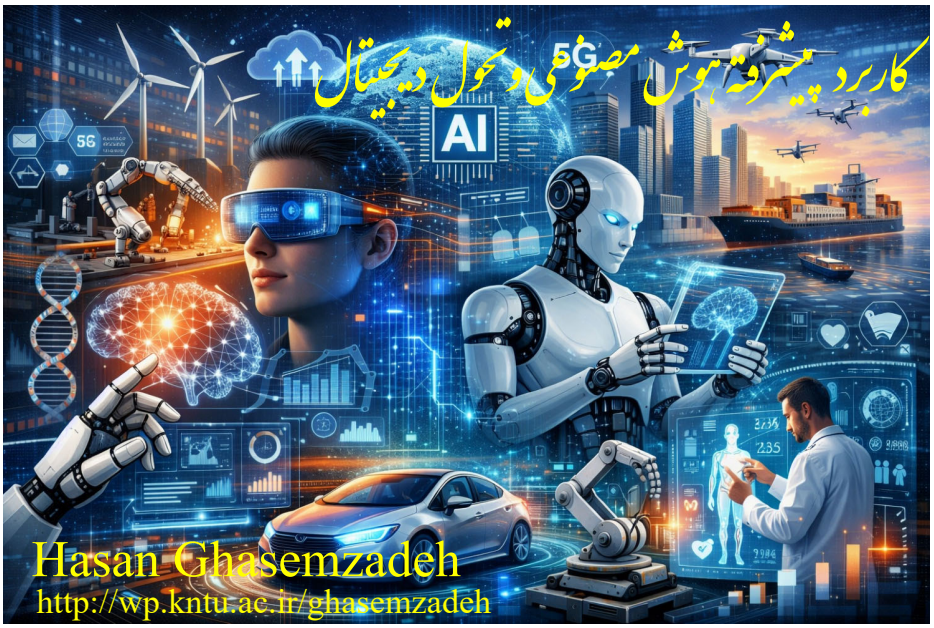
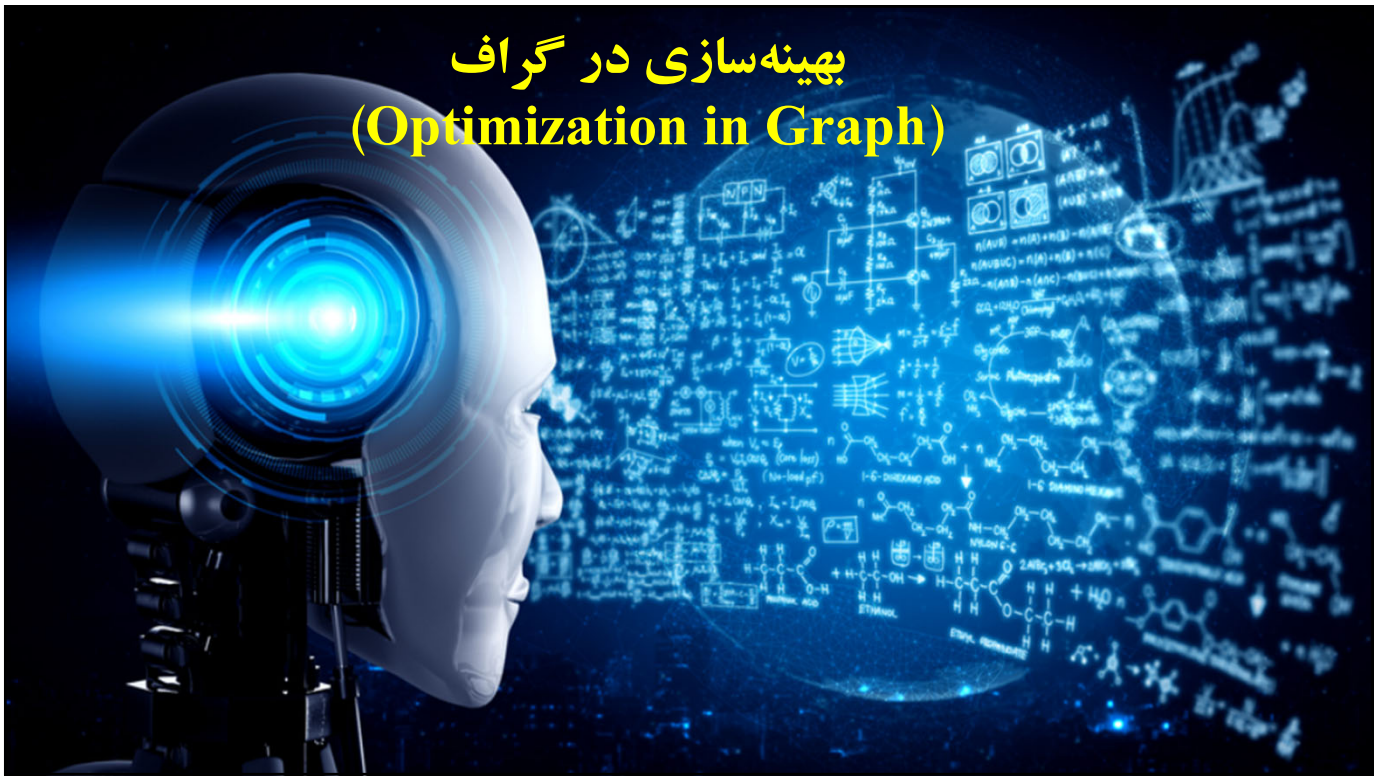
Advanced Application of Artificial Intelligence and Digital Transformation

K.N. Toosi University of Technology





فهرست مطالب- بهینه سازی در گراف

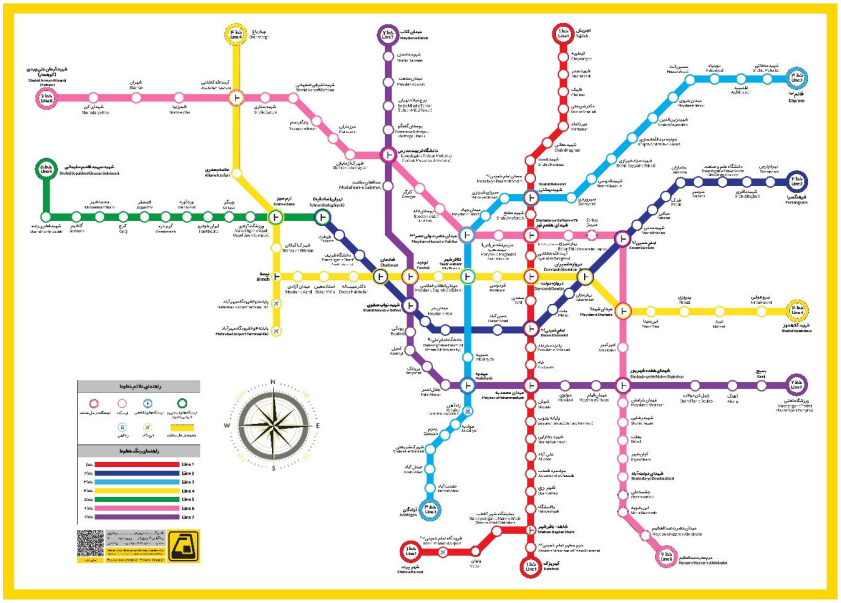
مقدمه
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

- Breadth-first Search
- Dijkstra's Method
- Bellman-Ford Method
- Floyd-Warshall Method

مقدمه

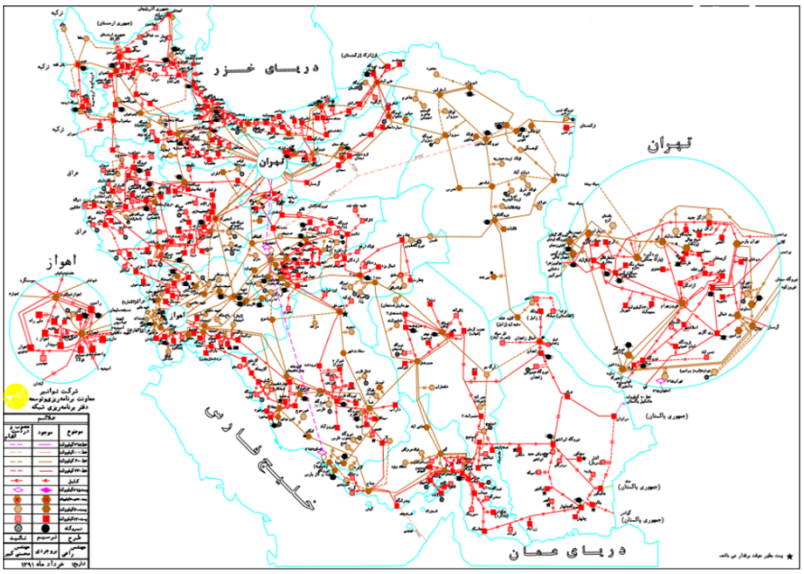
مقدمه

مقدمه



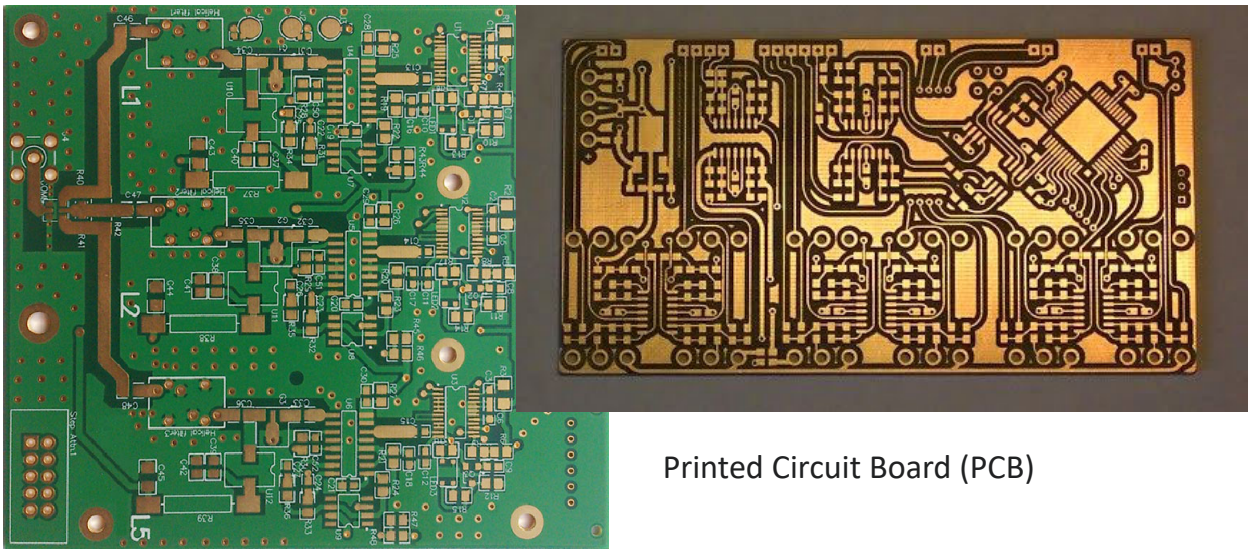
مقدمه

مقدمه



شبکه انتقال ۲۳۰ و ۴۰۰ کیلوولت ایران

مقدمه

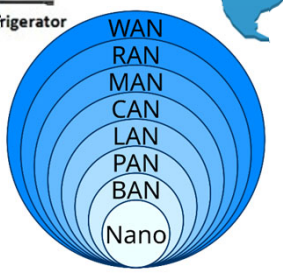


Printed Circuit Board (PCB)

مقدمه

Local Area Networks (LAN)

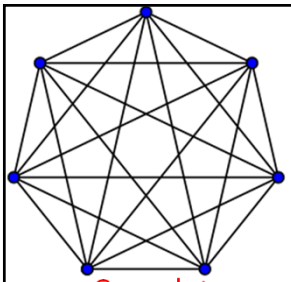
Wide area network (WAN)



Outline

تعریف گراف
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

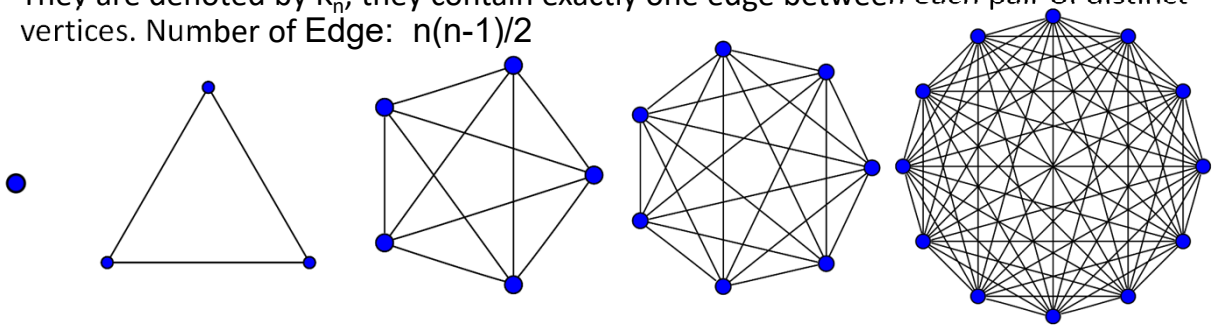
- Breadth-first Search
- Dijkstra's Method
- Bellman-Ford Method
- Floyd-Warshall Method



تعریف گراف

گراف از گره ها و یال ها تشکیل شده است
در گراف بدون جهت یال ها جهت ندارند و صرفا دو گره را به هم وصل می کنند.
گراف دارای انواع متنوعی است:

– Complete graph
They are denoted by K_n, they contain exactly one edge between each pair of distinct vertices. Number of Edge: n(n-1)/2

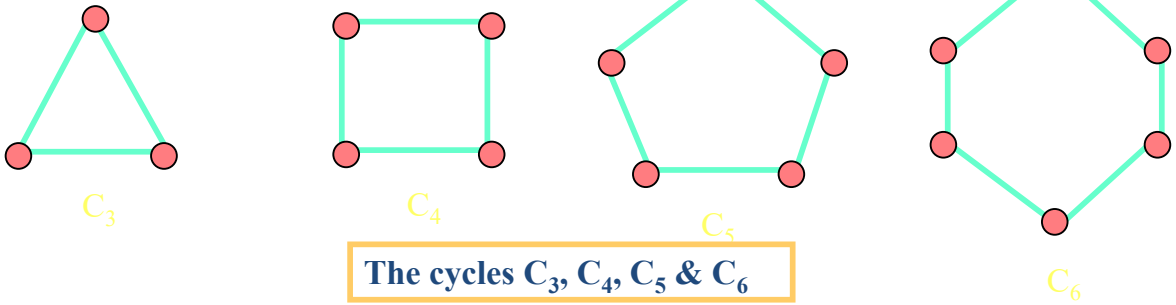


K_1, 0 K_3, 3 K_5, 10 K_7, 21 K_{12}, 66

تعریف گراف

- Cycles (ring)

They are denoted by $C_n (n \geq 3)$: they consist of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}$, $\{v_2, v_3\}, \ldots, \{v_n, v_{n-1}\}$ and $\{v_n, v_1\}$

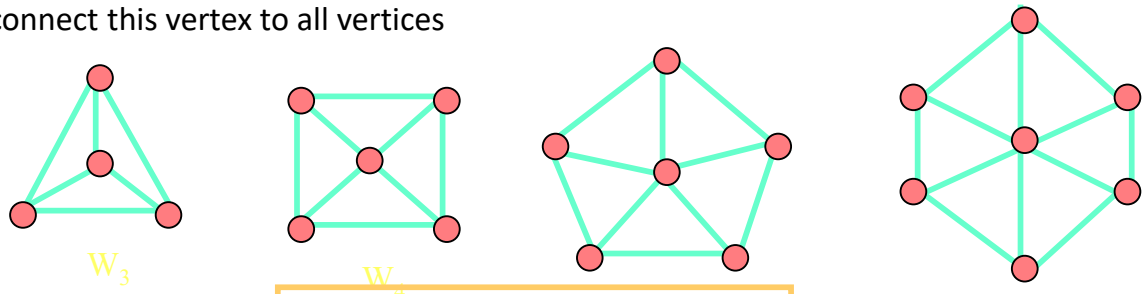


The cycles C_3, C_4, C_5 & C_6

تعریف گراف

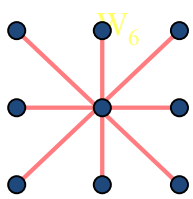
- Wheels

They are denoted by W_n; they are obtained by adding a vertex to the graphs C_n and connect this vertex to all vertices



The Wheels W_3, W_4, W_5 & W_6

star topology

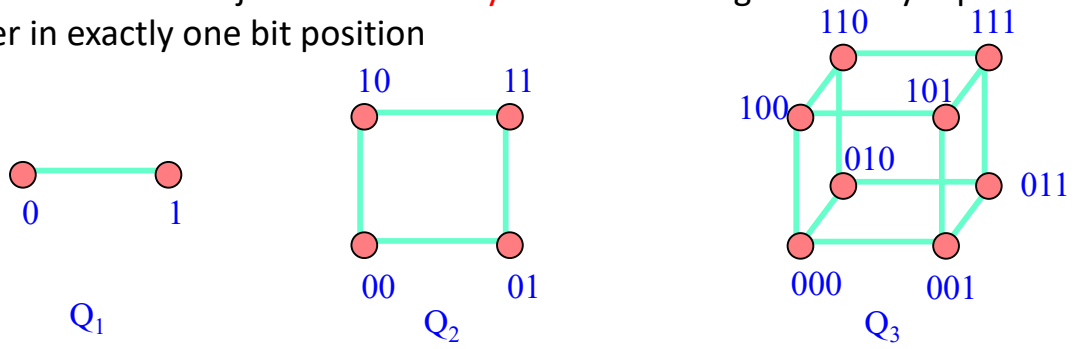


تعریف گراف

- n-cubes

  They are denoted by Q_n, they are graphs that have vertices representing the 2^n bit strings of length n.

  Two vertices are adjacent if and only if the bits strings that they represent differ in exactly one bit position



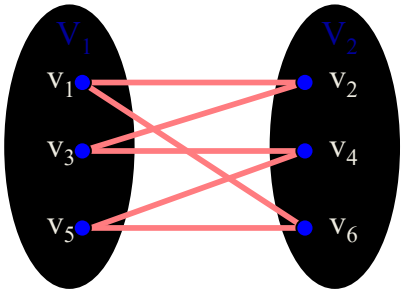
The n-cube Q_1 , Q_2 , Q_3

تعریف گراف

- Bipartite graph

  A simple graph is called bipartite if its vertex set V can be partitioned into 2 disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2 (so that no edge in G connects either 2 vertices in V_1 or 2 vertices in V_2).

  - Example: C_6 is bipartite, since its vertex set can be partitioned into the 2 sets $V_1 = \{v_1, v_3, v_5\}$ and $V_2 = \{v_2, v_4, v_6\}$, and every edge of C_6 connects a vertex in V_1 and a vertex in V_2.
  - Example: The star topology is equivalent to a $K_{1,n}$ complete bipartite graph
  - Example: K_3 is not bipartite. Why?

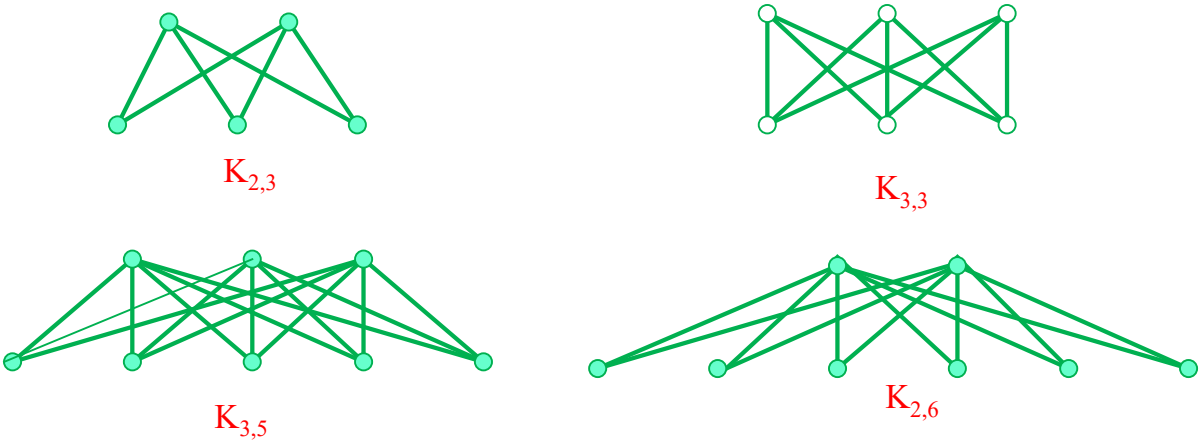


تعریف گراف

– Characterization of bipartite graph

- A graph is bipartite if and only if it is possible to color the vertices of the graph with at most 2 colors so that no 2 adjacent vertices have the same color
- Example: Complete bipartite graphs: they are denoted by $K_{m,n}$. Their vertices set is partitioned into 2 subsets of m and n vertices, respectively. There is an edge between 2 vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

تعریف گراف



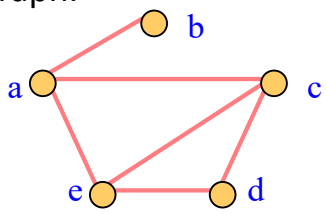
Some complete bipartite graphs

Representing Graphs & Graph Isomorphism

- Representing Graph
  - Goal: Consists of choosing the most convenient representation of a graph
  - We need to determine whether 2 graphs are isomorphic, this problem is important in graph theory
  - List all the edges of the graph (no multiple edges)
  - Use adjacency list, which specifies the vertices that are adjacent to each vertex of the graph

Representing Graphs & Graph Isomorphism (9.3) (cont.)

- Example: Use adjacency lists to describe this simple graph.



Solution:

Vertex	Adjacent vertices
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

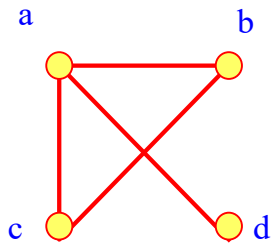
Representing Graphs & Graph Isomorphism (9.3) (cont.)

- Adjacency matrices
  - To simplify computation, graphs can be represented using matrices
    - Adjacency matrix
    - Incident matrix
  - The adjacency matrix is defined as $A = [a_{ij}]$ such that

$$a_{ij} = \begin{cases} 1 \text{ if } \{v_i, v_j\} \text{ is an edge of } G \\ 0 \text{ otherwise} \end{cases}$$

Representing Graphs & Graph Isomorphism (9.3) (cont.)

- Example: Use a adjacency matrix to represent this graph:

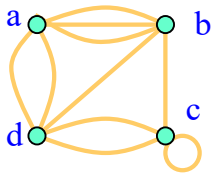


Solution: We order the vertices a, b, c, d.
The matrix representing this graph is

	e_a	e_b	e_c	e_d
$a = v_1$	0	1	1	1
$b = v_2$	1	0	1	0
$c = v_3$	1	1	0	0
$d = v_4$	1	0	0	0

Representing Graphs & Graph Isomorphism (9.3) (cont.)

- In case of pseudographs , the adjacency matrix is not a binary matrix but is formed of elements that represent the number of edges between 2 vertices
- Example: Use an adjacency matrix to represent this pseudograph:



Solution: The adjacency matrix using the ordering of vertices a, b, c, d is:

$$\begin{array}{c} \\ a=v_1 \\ b=v_2 \\ c=v_3 \\ d=v_4 \end{array} \begin{array}{c} \begin{array}{cccc} e_a & e_b & e_c & e_d \end{array} \\ \begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix} \end{array}$$

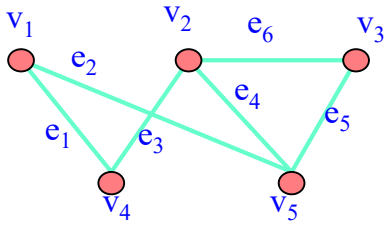
Representing Graphs & Graph Isomorphism (9.3) (cont.)

- Incidence matrices
  - Let G = (V,E) be an undirected graph.
  - Incidence matrices are defined by the matrix M = [m_{ij}] such that

$$m_{ij} = \begin{cases} 1 & \text{if edge } e_j \text{ is incident with vertex } v_i \\ 0 & \text{otherwise} \end{cases}$$

Representing Graphs & Graph Isomorphism (9.3) (cont.)

– Example: Using an incidence matrix, represent the following undirected graph:



Solution:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

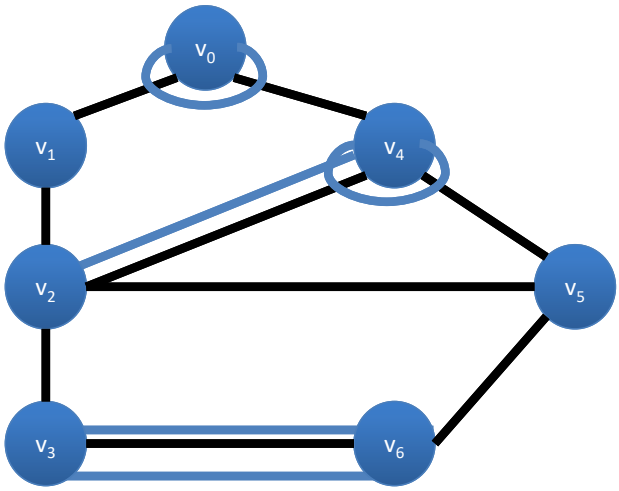
Outline

تعریف گراف
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

- Breadth-first Search
- Dijkstra's Method
- Bellman-Ford Method
- Floyd-Warshall Method

Parallel Edges and Loop

G = (V, E)



Parallel Edges Represented by the same pair of vertices.

Loop

Edges that connect a vertex to itself.

Simple Graphs

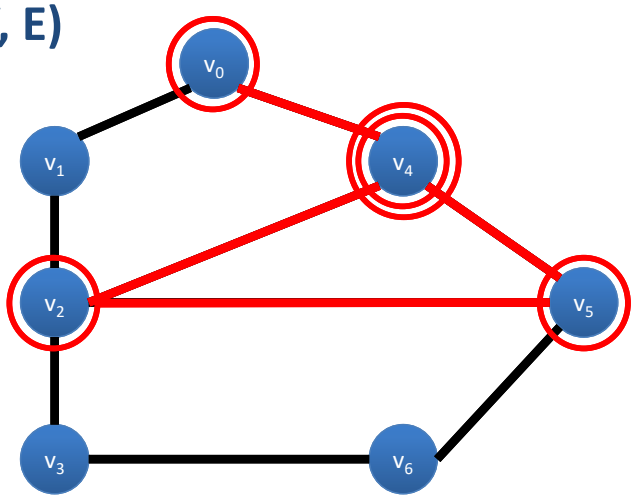
Graphs without parallel edges and without loops.

Degree of a Vertex

Number of edges incident on the vertex ($deg(v_1) = 2$, $deg(v_5) = 3$)

Walk(پیمایش)

G = (V, E)

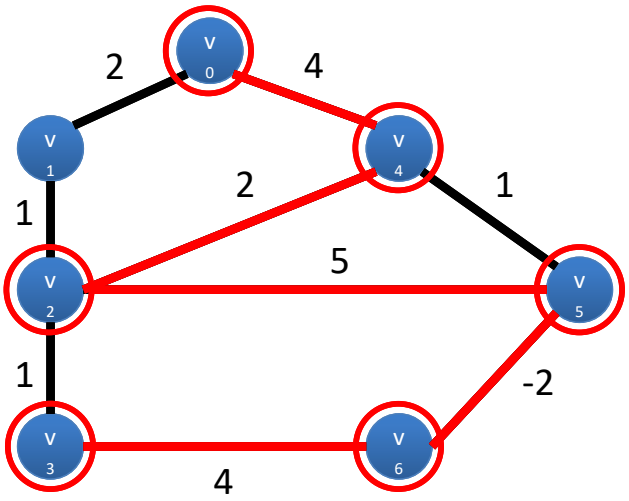


Length of walk= 4
(Number of edges)

Sequence $P = (v_0,e_1,v_1,\ldots,e_k,v_k)$, $e_i = (v_{i-1},v_i)$

v_0, (v_0,v_4), v_4, (v_4,v_2), v_2, (v_2,v_5), v_5, (v_5,v_4), v_4

Length of a Walk



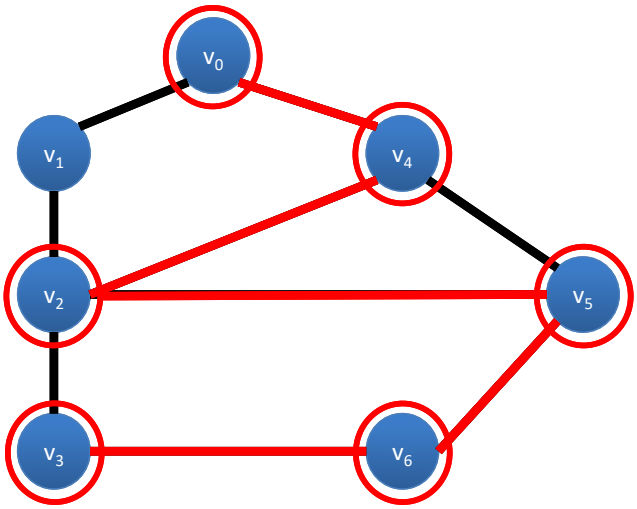
G = (V, E)

l: E ➔ R

Sum of lengths of all edges: $\Sigma_i\ l(e_i)$

Length of above walk = 4+2+5-2+4 = 13

Path(مسیر)



G = (V, E)

Sequence $P = (v_0,e_1,v_1,\ldots,e_k,v_k)$, $e_i = (v_{i-1},v_i)$

Vertices $v_0,v_1,\ldots,v_k$ are distinct

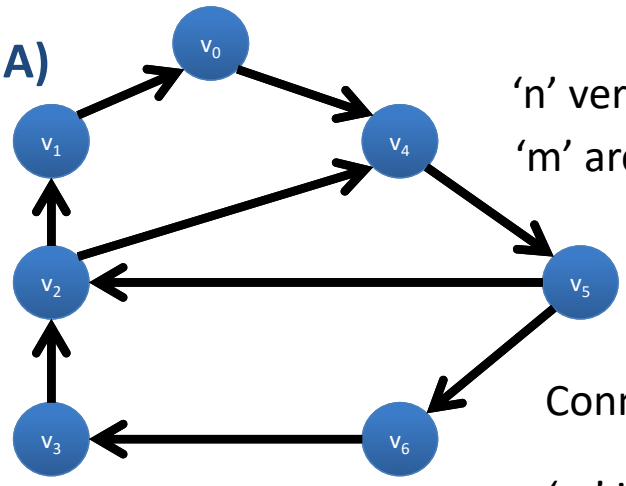
Outline

تعریف گراف
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

- Breadth-first Search
- Dijkstra's Method
- Bellman-Ford Method
- Floyd-Warshall Method

Directed Graphs (Digraphs)

D = (V, A)



'n' vertices or nodes V

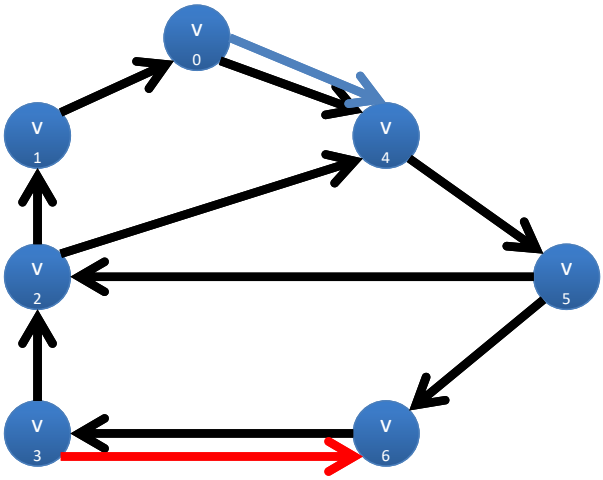
'm' arcs A: ordered pairs from V

Connected by an arc a = (u,v).

'v_0' is the inneighbor of 'v_4'

'v_4' is the outneighbor of 'v_0'

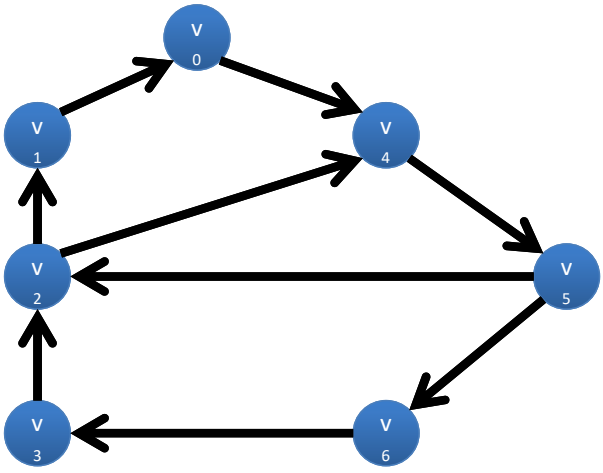
Parallel Arcs



D = (V, A)

Represented by the same ordered pair of vertices.

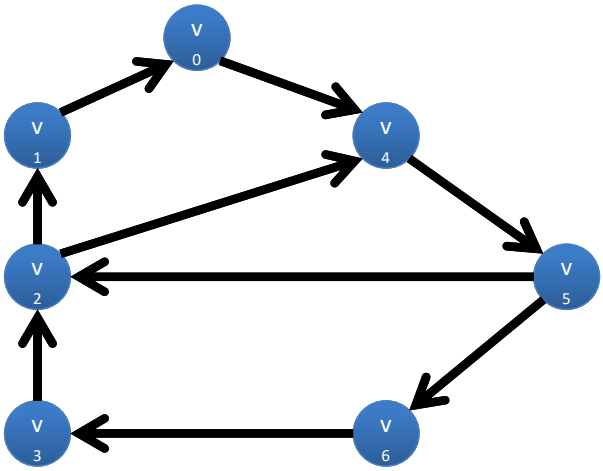
Simple Graphs



D = (V, A)

Graphs without parallel arcs and without loops.

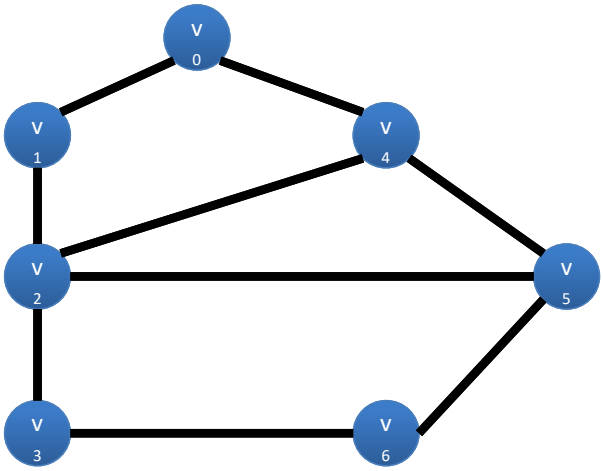
Underlying Undirected Graph



D = (V, A)

Graphs obtained by ignoring orientation of arcs.

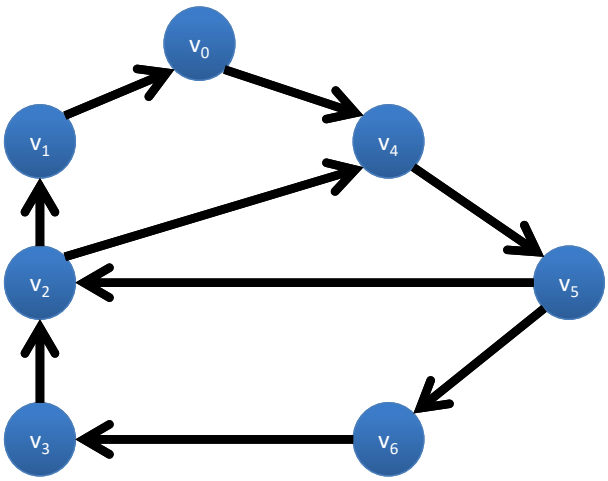
Underlying Undirected Graph



G = (V, E)

Graphs obtained by ignoring orientation of arcs.

Indegree of a Vertex

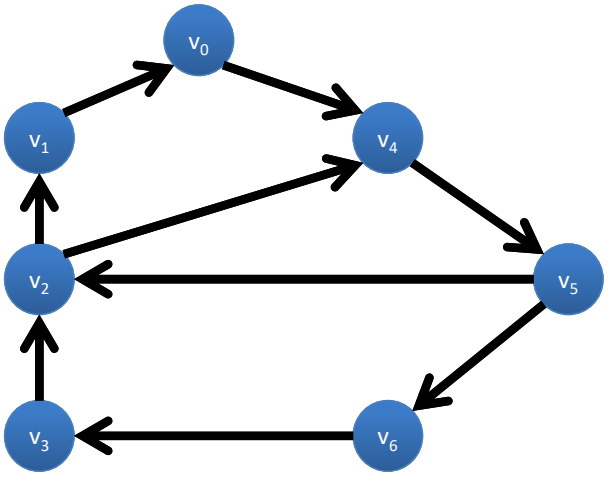


D = (V, A)

Number of arcs entering the vertex.

$indeg(v_0) = 1$, $indeg(v_1) = 1$, $indeg(v_4) = 2$, …

Outdegree of a Vertex



D = (V, A)

Number of arcs leaving the vertex.

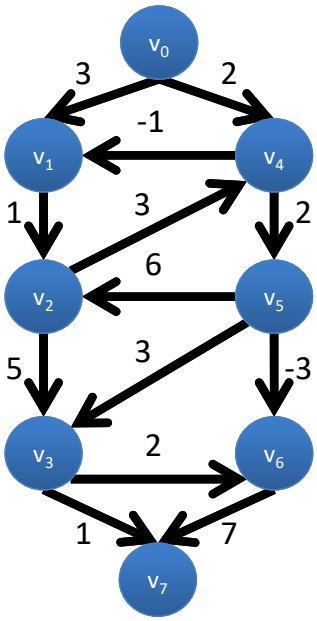
$outdeg(v_0) = 1$, $outdeg(v_1) = 1$, $outdeg(v_2) = 2$, …

Outline

تعریف گراف
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

- Breadth-first Search
- Dijkstra's Method
- Bellman-Ford Method
- Floyd-Warshall Method
- A* (A-Star) Method
- Johnson Method
- Bidirectional)BFS(Method
- Genetic Algorithm (GA)
- Ant Colony Optimization (ACO)

The Shortest Path Problem

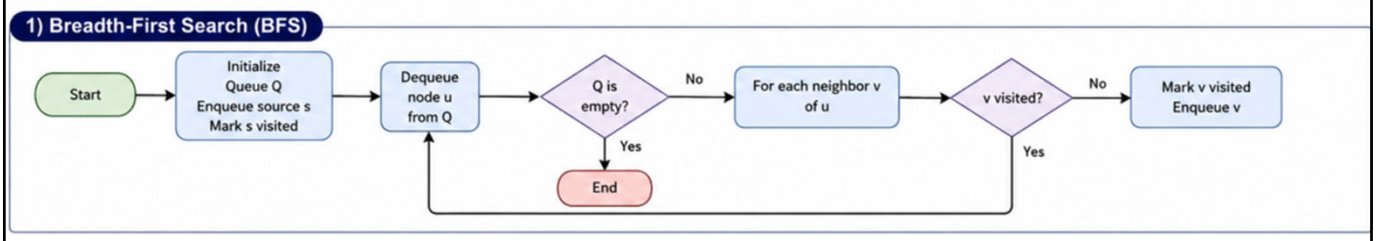


Find the shortest path from s to t

Length of path =
Σ Length of arcs

Breadth-first Search

الگوریتم BFSجستجو را از رأس مبدأ آغاز می‌کند و ابتدا تمام همسایه‌های مستقیم آن را بررسی می‌کند، سپس همسایه‌های همسایه‌ها را و به همین ترتیب لایه به لایه در گراف پیش می‌رود. برای این کار از یک **صف (Queue)** استفاده می‌شود تا رأس‌ها به ترتیب ورود پردازش شوند. از آنجا که تمام رأس‌های واقع در فاصله ۱، سپس فاصله ۲ و ... بررسی می‌شوند، BFS در گراف‌های بدون وزن کوتاه‌ترین مسیر را بر حسب تعداد یال‌ها پیدا می‌کند.



کاربرد	پیچیدگی زمانی	معایب	مزایا
شبکه‌های اجتماعی، جستجوی وب، مسیریابی در گراف‌های بدون وزن، بازی‌ها	(O(V+E))	برای گراف‌های وزن‌دار مناسب نیست	ساده و سریع، تضمین یافتن کوتاه‌ترین مسیر در گراف بدون وزن، حافظه کم

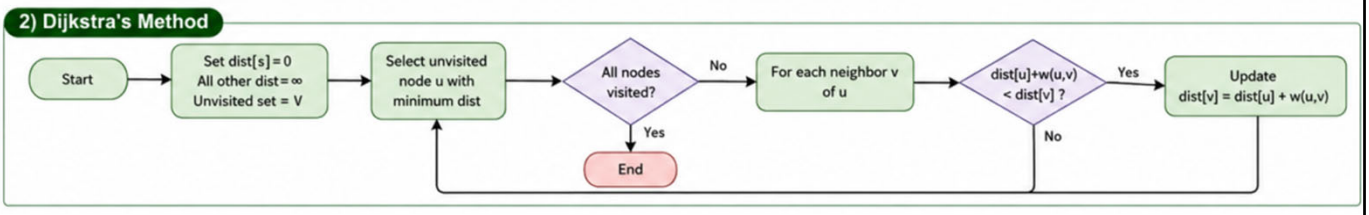
Outline

تعریف گراف
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

- Breadth-first Search
- Dijkstra's Method
- Bellman-Ford Method
- Floyd-Warshall Method
- A* (A-Star) Method
- Johnson Method
- Bidirectional)BFS(Method
- Genetic Algorithm (GA)
- Ant Colony Optimization (ACO)

Dijkstra's Method

الگوریتم دایکسترا برای یافتن کوتاه‌ترین مسیر از یک مبدأ به تمام رأس‌های دیگر در گراف‌های دارای وزن مثبت استفاده می‌شود. در هر مرحله، رأسی که کمترین فاصله موقت از مبدأ را دارد انتخاب می‌شود و فاصله همسایگان آن به‌روزرسانی می‌گردد (عمل Relaxation). این فرآیند تا زمانی ادامه می‌یابد که فاصله همه رأس‌ها قطعی شود. ایده اصلی دایکسترا این است که وقتی رأسی با کمترین فاصله انتخاب شد، دیگر مسیر کوتاه‌تری برای آن وجود نخواهد داشت.



کاربرد	پیچیدگی زمانی	معایب	مزایا
GPS، مسیریابی جاده‌ای، شبکه‌های کامپیوتری، سیستم‌های حمل‌ونقل	$O(E \log V)$ با Heap	با وزن‌های منفی کار نمی‌کند	کوتاه‌ترین مسیر را با دقت بالا پیدا می‌کند، برای شبکه‌های وزن‌دار مثبت بسیار کارآمد است

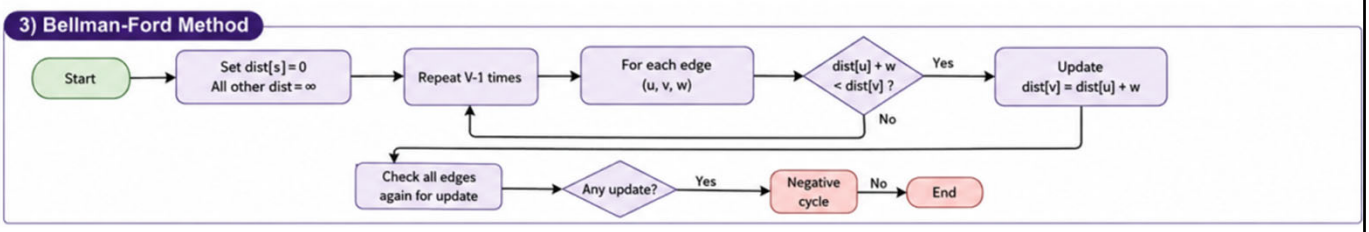
Outline

تعریف گراف
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

- Breadth-first Search
- Dijkstra's Method
- Bellman-Ford Method
- Floyd-Warshall Method
- A* (A-Star) Method
- Johnson Method
- Bidirectional)BFS(Method
- Genetic Algorithm (GA)
- Ant Colony Optimization (ACO)

Bellman-Ford Method

الگوریتم بلمن–فورد نیز کوتاه‌ترین مسیر از یک مبدأ را پیدا می‌کند، اما برخلاف دایکسترا قادر به کار با یال‌های دارای وزن منفی است. روش کار آن این است که تمام یال‌های گراف به تعداد $V-1$ بار (تعداد رأس‌ها منهای یک) بررسی و Relax می‌شوند. با این کار تأثیر مسیرهای طولانی‌تر به تدریج در فاصله‌ها منعکس می‌شود. پس از پایان تکرارها، یک بار دیگر همه یال‌ها بررسی می‌شوند؛ اگر هنوز بتوان فاصله‌ای را کاهش داد، وجود چرخه با وزن منفی تشخیص داده می‌شود



کاربرد	پیچیدگی زمانی	معایب	مزایا
پروتکل‌های مسیریابی مانند RIP، تحلیل اقتصادی و مالی، شبکه‌هایی با هزینه‌های منفی	$(O(VE))$	کندتر از دایکسترا است	وزن‌های منفی را پشتیبانی می‌کند، چرخه‌های منفی را تشخیص می‌دهد

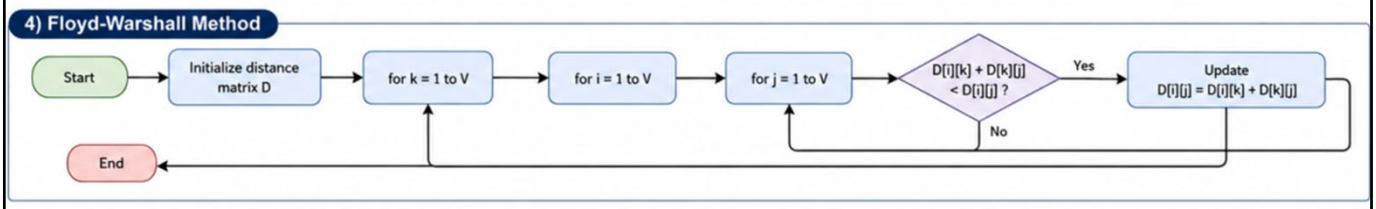
Outline

تعریف گراف
انواع گراف
گراف بدون جهت
گراف جهت دار
الگوریتم های یافتن مسیر بهینه

- Breadth-first Search
- Dijkstra’s Method
- Bellman-Ford Method
- Floyd-Warshall Method
- A* (A-Star) Method
- Johnson Method
- Bidirectional)BFS(Method
- Genetic Algorithm (GA)
- Ant Colony Optimization (ACO)

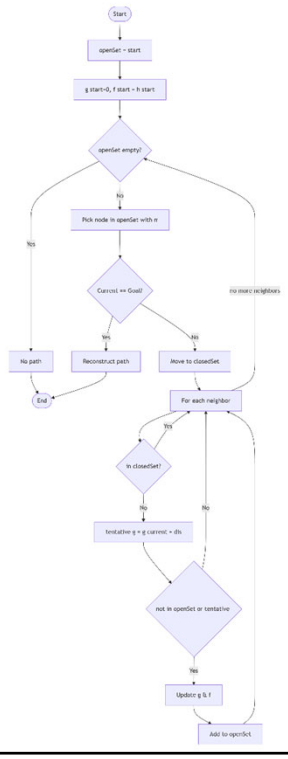
Floyd-Warshall Method

الگوریتم فلوید–وارشال کوتاه‌ترین مسیر بین تمام جفت رأس‌های گراف را به طور همزمان محاسبه می‌کند. این روش با تشکیل یک ماتریس فاصله شروع می‌شود و سپس به صورت سیستماتیک بررسی می‌کند که آیا عبور از یک رأس واسطه k می‌تواند مسیر بین دو رأس i و j را کوتاه‌تر کند یا خیر. اگر مسیر $i \to k \to j$ کوتاه‌تر باشد، مقدار ماتریس به‌روزرسانی می‌شود. پس از بررسی تمام رأس‌های واسطه، ماتریس نهایی شامل کوتاه‌ترین فاصله بین هر دو رأس گراف خواهد بود.



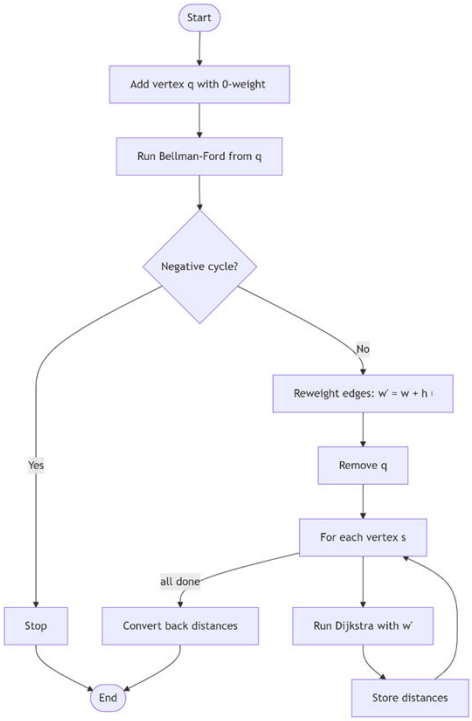
کاربرد	پیچیدگی زمانی	معایب	مزایا
تحلیل شبکه‌های کوچک تا متوسط، حمل‌ونقل شهری، ماتریس فاصله بین تمام نقاط	$(O(V^3))$	حافظه و زمان زیاد برای گراف‌های بزرگ	کوتاه‌ترین مسیر بین همه جفت رأس‌ها را محاسبه می‌کند، پیاده‌سازی ساده

A* (A-Star) Method



الگوریتم A* نسخه هوشمند دایکسترا است. علاوه بر هزینه طی‌شده از مبدأ $g(n)$، از یک تابع تخمین فاصله تا مقصد $h(n)$ نیز استفاده می‌کند و گرهی را انتخاب می‌کند که مقدار $f(n)=g(n)+h(n)$ آن کمینه باشد. اگر تابع تخمین مناسب باشد، A* بسیار سریع‌تر از دایکسترا به مقصد می‌رسد و همچنان کوتاه‌ترین مسیر را تضمین می‌کند.

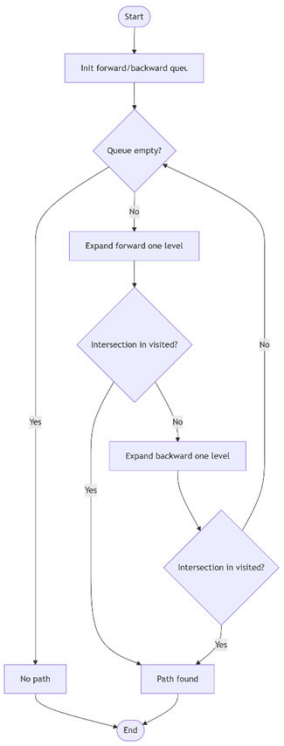
کاربرد	پیچیدگی زمانی	معایب	مزایا
ناوبری ربات‌ها، بازی‌های رایانه‌ای، GPS پیشرفته	حدود $O(E)$ تا نمایی	کیفیت وابسته به Heuristic است	بسیار سریع‌تر از دایکسترا در صورت داشتن تابع Heuristic مناسب



Johnson Method

الگوریتم جانسون برای یافتن کوتاه‌ترین مسیر بین همه جفت رأس‌ها در گراف‌های بزرگ و تنک طراحی شده است. ابتدا با اجرای Bellman-Ford وزن یال‌ها را بازتعریف می‌کند تا همه وزن‌ها غیرمنفی شوند. سپس از هر رأس یک بار الگوریتم دایکسترا اجرا می‌شود. این روش نسبت به Floyd-Warshall برای گراف‌های بزرگ بسیار سریع‌تر است.

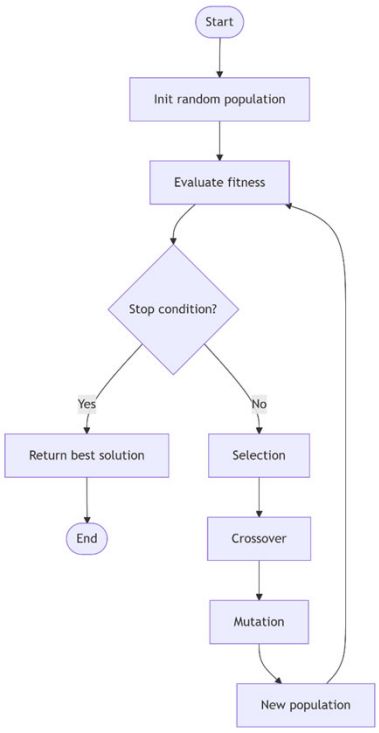
کاربرد	پیچیدگی زمانی	معایب	مزایا
شبکه‌های بزرگ و تنک	$(O(VE + V^2 \log V))$	پیاده‌سازی پیچیده‌تر	برای گراف‌های تنک و همه‌جفت‌مسیرها مناسب، وزن منفی را پشتیبانی می‌کند



Bidirectional (BFS) Method

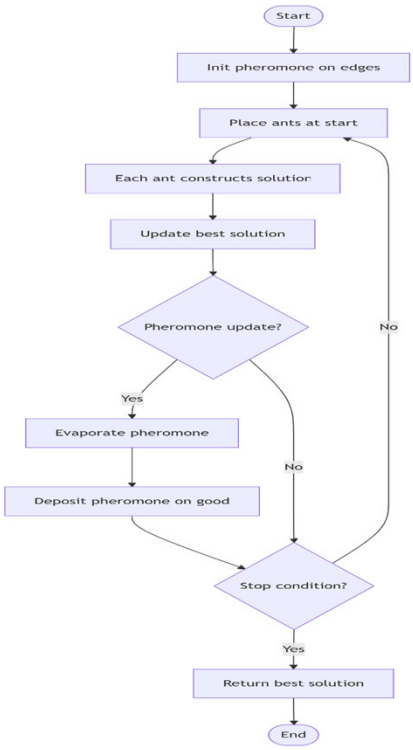
در این روش به جای جستجو فقط از مبدأ، دو جستجوی BFS همزمان از مبدأ و مقصد آغاز می‌شوند. هنگامی که دو جبهه جستجو به هم برسند، مسیر کامل ساخته می‌شود. چون عمق جستجو تقریباً نصف می‌شود، تعداد گره‌های بررسی‌شده به شدت کاهش می‌یابد و سرعت افزایش می‌یابد.

کاربرد	پیچیدگی زمانی	معایب	مزایا
مسیریابی در شبکه‌های بزرگ بدون وزن	حدود $O(b^{d/2})$	نیاز به دانستن مبدأ و مقصد دارد	تقریباً ریشه دوم BFS از نظر تعداد گره‌های بازدیدشده



Genetic Algorithm (GA)

الگوریتم ژنتیک از فرآیند تکامل زیستی الهام گرفته است. ابتدا مجموعه‌ای از جواب‌های تصادفی (جمعیت) تولید می‌شود. سپس بهترین جواب‌ها انتخاب شده، با هم ترکیب (Crossover) می‌شوند و تغییرات تصادفی (Mutation) روی آنها اعمال می‌شود. این فرآیند نسل به نسل تکرار می‌شود تا پاسخ‌های بهتری تولید شوند. در مسائل پیچیده بهینه‌سازی که فضای جستجو بسیار بزرگ است کاربرد فراوانی دارد.



Ant Colony Optimization (ACO)

الگوریتم کلونی مورچگان از رفتار مورچه‌ها در یافتن کوتاه‌ترین مسیر الهام گرفته است. مورچه‌های مصنوعی روی گراف حرکت می‌کنند و روی مسیرهای طی‌شده فرومون مجازی برجا می‌گذارند. مسیرهای بهتر فرومون بیشتری دریافت می‌کنند و احتمال انتخاب آنها در تکرارهای بعدی افزایش می‌یابد. با گذشت زمان، جمعیت مورچه‌ها به سمت مسیرهای بهینه همگرا می‌شود. این روش در مسائل مسیریابی، زمان‌بندی و بهینه‌سازی ترکیبی بسیار موفق بوده است.

مقایسه روشها

الگوریتم	مزایا	معایب	پیچیدگی زمانی	کاربرد
Breadth-First Search (BFS)	ساده و سریع، تضمین یافتن کوتاه‌ترین مسیر در گراف بدون وزن، حافظه کم	برای گراف‌های وزن‌دار مناسب نیست	(O(V+E))	شبکه‌های اجتماعی، جستجوی وب، مسیریابی در گراف‌های بدون وزن، بازی‌ها
Dijkstra	کوتاه‌ترین مسیر را با دقت بالا پیدا می‌کند، برای شبکه‌های وزن‌دار مثبت بسیار کارآمد است	با وزن‌های منفی کار نمی‌کند	O(E(log V))	GPS، مسیریابی جاده‌ای، شبکه‌های کامپیوتری، سیستم‌های حمل‌ونقل
Bellman-Ford	وزن‌های منفی را پشتیبانی می‌کند، چرخه‌های منفی را تشخیص می‌دهد	کندتر از دایکسترا است	(O(VE))	پروتکل‌های مسیریابی مانند RIP، تحلیل اقتصادی و مالی، شبکه‌هایی با هزینه‌های منفی
Floyd-Warshall	کوتاه‌ترین مسیر بین همه جفت رأس‌ها را محاسبه می‌کند، پیاده‌سازی ساده	حافظه و زمان زیاد برای گراف‌های بزرگ	(O(V^3))	تحلیل شبکه‌های کوچک تا متوسط، حمل‌ونقل شهری، ماتریس فاصله بین تمام نقاط
A*	بسیار سریع‌تر از دایکسترا در صورت داشتن تابع Heuristic مناسب	کیفیت وابسته به Heuristic است	حدود O(E) تا نمایی	ناوبری ربات‌ها، بازی‌های رایانه‌ای، GPS پیشرفته
Johnson	برای گراف‌های تنک و همه‌جفت‌مسیرها مناسب، وزن منفی را پشتیبانی می‌کند	پیاده‌سازی پیچیده‌تر	(O(VE + V²log V))	شبکه‌های بزرگ و تنک
Bidirectional BFS	تقریباً ریشه دوم BFS از نظر تعداد گره‌های بازدیدشده	نیاز به دانستن مبدأ و مقصد دارد	حدود $O(b^{d/2})$	مسیریابی در شبکه‌های بزرگ بدون وزن

Time Complexity

مقایسه مقیاس‌پذیری

برای گرافی با $V = 10^5, E = 10^6$ تقریباً:

الگوریتم	تعداد عملیات
BFS	(10^6)
Dijkstra	(10^7)
Bellman-Ford	(10^{11})
Floyd-Warshall	(10^{15})

برنامه مقایسه الگوریتمهای مختلف

در یک گراف با پانزده گره از الگوریتم های مختلف برای بهینه کردن مسیر بر حسب طول ، زمان، سوخت، قابلیت اعتماد و ترکیبی از آنها در سفر استفاده کنید

معیارها Distance / Time / Fuel / Reliability و تابع هدف چندمعیاره با وزن‌دهی

```
cases = [

("SHORTEST PATH",(1,0,0,0)),
("FASTEST PATH",(0,1,0,0)),
("MIN FUEL PATH",(0,0,1,0)),
("MOST RELIABLE PATH",(0,0,0,1)),
("MULTI OBJECTIVE",(0.35,0.35,0.2,0.1))

]
```

الگوریتم‌ها :
Dijkstra
A*
Bellman-Ford
Genetic Algorithm (GA)
Ant Colony Optimization (ACO)

برنامه 16 graph best way comparison.py

کتابخانه‌ها

Graph Analysis & Classic Optimization

کتابخانه	سازنده	الگوریتم‌های کلیدی	مناسب برای
NetworkX	جامعه متن‌باز	Dijkstra ،A* ،Bellman-Ford ،MST ،TSP ، PageRank ،Community Detection	تحلیل شبکه، تحقیقات، نمونه‌سازی
igraph	igraph team	کوتاه‌ترین مسیر، خوشه‌بندی، مرکزیت، تشخیص جامعه	تحلیل شبکه‌های اجتماعی، بیولوژی
NetworKit	KIT + HU Berlin	مرکزیت، خوشه‌بندی، مسیریابی موازی	گراف‌های میلیاردی، پردازش موازی
Graphillion	JST Japan	جستجو، بهینه‌سازی و شمارش روی مجموعه‌ای از گراف‌ها	ارزیابی شبکه برق، تحلیل راه‌آهن

Graph Nural Networks (GNN)

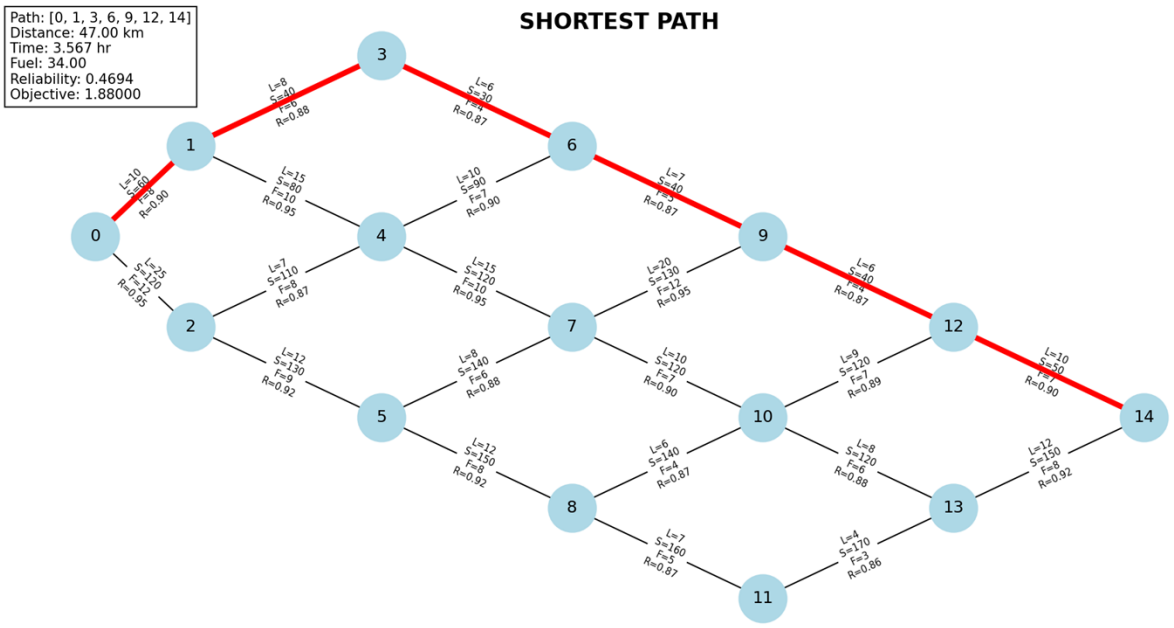
کتابخانه	سازنده	مدل‌های پشتیبانی‌شده	مناسب برای
PyTorch Geometric (PyG)	Fey & Lenssen / Stanford	GCN ،GAT ،GraphSAGE ،GIN ،MPNN	کشف دارو، شبکه اجتماعی، مولکول‌ها
DGL (Deep Graph Library)	AWS / Carnegie Mellon	GraphSAGE ،GAT ،GIN ،RGCN ،HGT	گراف‌های ناهمگن، بیوانفورماتیک، AlphaFold
Spektral	جامعه متن‌باز	GCN ،GAT ،APPNP ،GraphSAGE	کاربران TensorFlow/Keras
Jraph	Google DeepMind	با تابع پیام‌رسانی GNNپیاده‌سازی انعطاف‌پذیر	تحقیقات، اکوسیستم JAX

کتابخانه‌ها

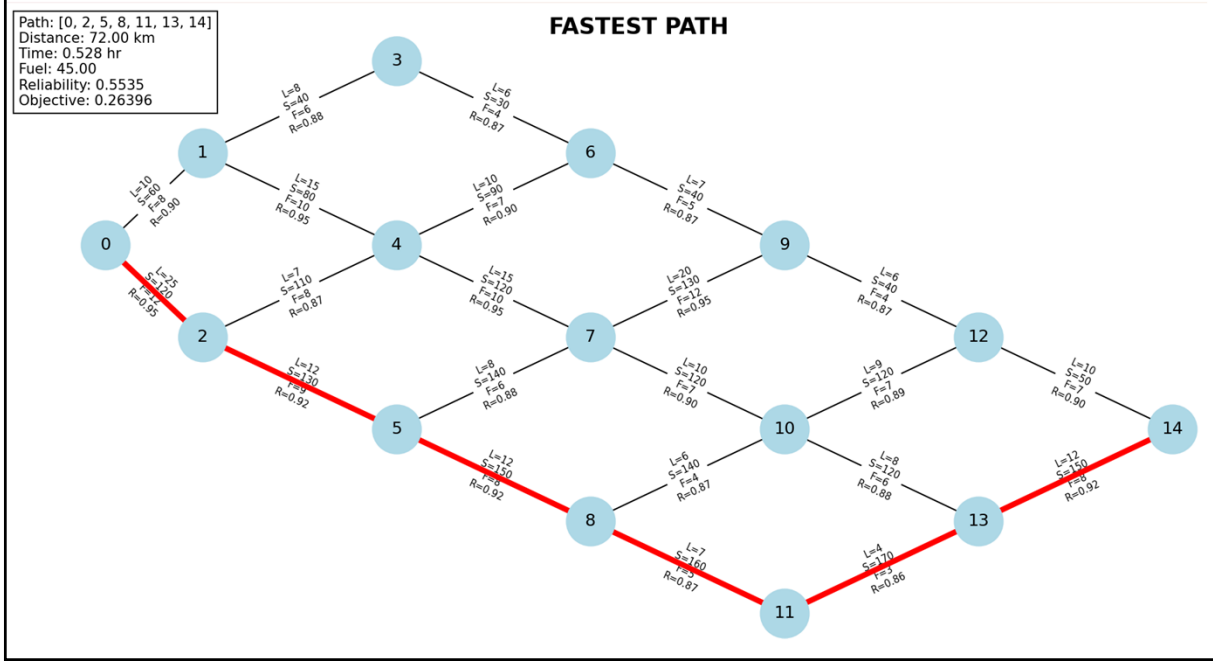
راهنمای انتخاب

بهترین کتابخانه	نوع مسئله
NetworkX	تحلیل گراف کوچک–متوسط، تحقیقات
Google OR-Tools	مسیریابی، TSP، VRP، لجستیک
NetworKit	گراف‌های میلیاردی، پردازش موازی CPU
RAPIDS cuGraph	گراف‌های بزرگ روی GPU
PyTorch Geometric	یادگیری عمیق روی گراف (PyTorch)
DGL	یادگیری عمیق روی گراف (چند فریمورک)
SnapVX	بهینه‌سازی محدب روی گراف
PyKEEN	گراف دانش و تکمیل دانش

برنامه مقایسه الگوریتمهای مختلف

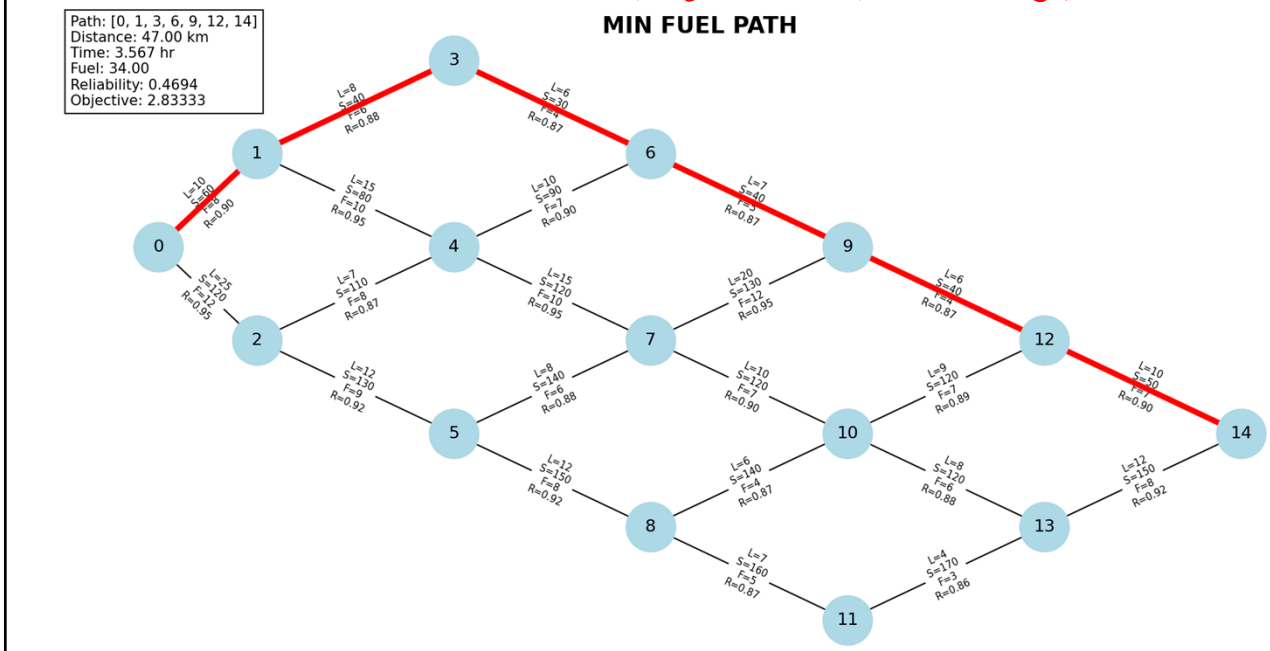


برنامه مقایسه الگوریتمهای مختلف



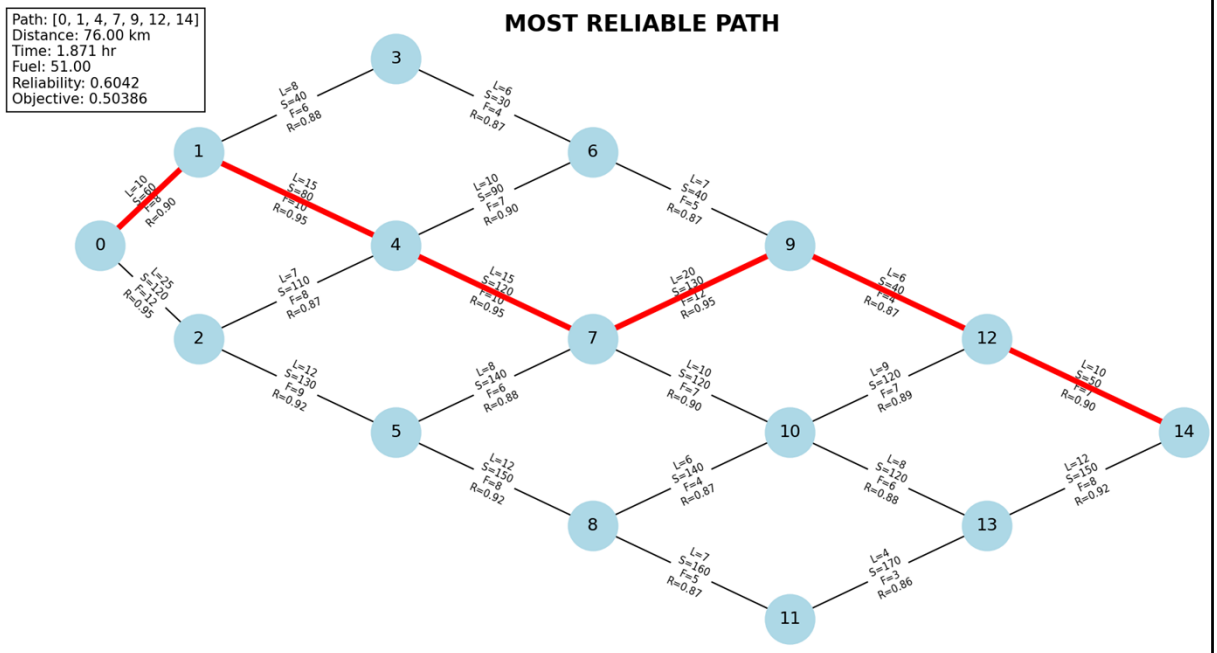
59

برنامه مقایسه الگوریتمهای مختلف

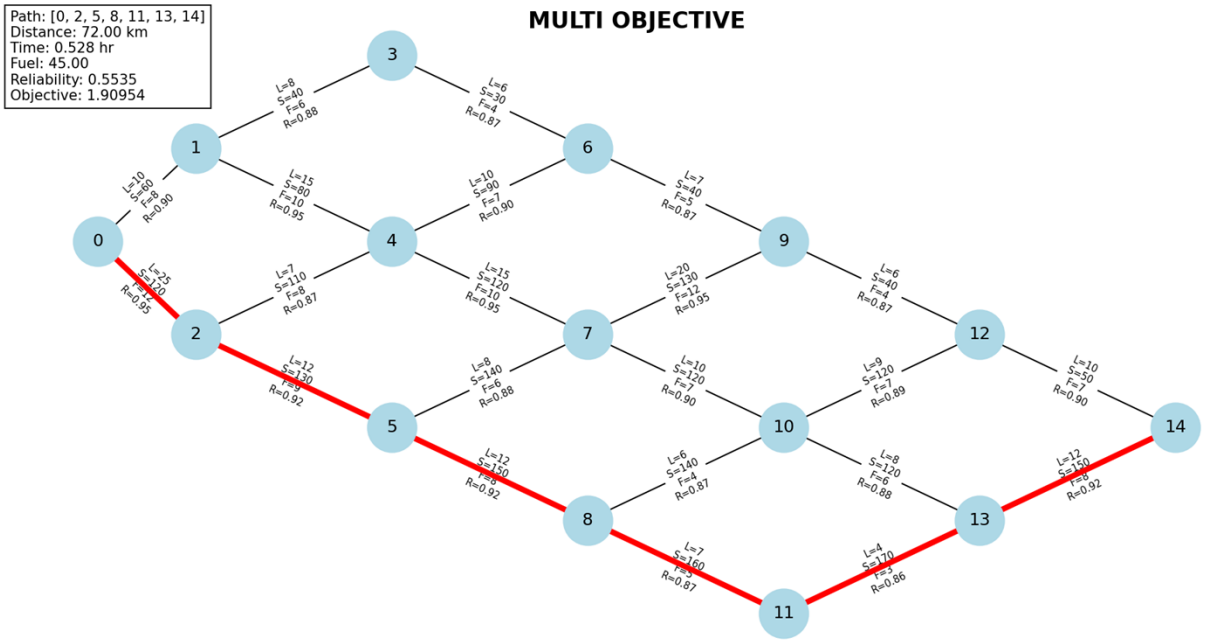


60

برنامه مقایسه الگوریتمهای مختلف



برنامه مقایسه الگوریتمهای مختلف



62

نمونه استفاده از گراف در عمران

A spatiotemporal displacement prediction method for InSAR-detected landslides using a graph neural network coupling spatial and temporal features.

Zhang, Y., He, Y., Gao, F., Huo, T., Zhang, Q., Lu, J., & Zhang, L. (2025). *Geomatics, Natural Hazards and Risk*, *16*(1).

یک روش مبتنی بر GNNمکانی-زمانی برای پیش‌بینی جابجایی رانش‌های زمین شناسایی‌شده با InSARتوسعه داده شد که روابط توپولوژیکی مکانی را با ویژگی‌های زمانی پیوند می‌دهد. این روش، رانش زمین را به صورت یک نمودار از دیدگاه ژئومکانیکی نشان می‌دهد، که در آن ساختار زمین‌شناسی و شرایط محیطی، اتصال گره‌ها را کنترل می‌کنند.

نمونه استفاده از گراف در عمران

Controllable and flexible residential floor plan layout design based on multi-agent deep reinforcement learning with layout prior size and similar experience abandon,

Gan Luo, Xuhong Zhou, Liang Feng, Jiepeng Liu, Pengkun Liu, Yunzhu Liao, Wenchen Shan, Hongtuo Qi, Advanced Engineering Informatics, Volume 68, Part B, 2025,

یک چارچوب GNN-GANبرای مدل‌سازی چینه‌شناسی زیرسطحی با استفاده از شبکه‌های توجه گراف (GAT) و GraphSAGE توسعه داده شد که داده‌های CPTو داده‌های گمانه را در یک سایت خط مترو سوژو در چین در بر می‌گیرد.

نمونه استفاده از گراف در عمران

Three-dimensional voxel geological modelling for subsurface stratigraphy: a graph convolutional network approach

Wang, L., Pan, Q., Su, D., & Huang, S. (2025). Canadian Geotechnical Journal, 62, 1–15.

یک نمودار توپولوژیکی با نقاط مکانی به عنوان گره ساخته شده و انواع خاک و مختصات مکانی به عنوان ویژگی‌های گره کدگذاری می‌شوند و همبستگی‌های مکانی از طریق یال‌های وزن‌دار تعیین می‌شوند.

نمونه استفاده از گراف در عمران

Graph Neural Network-based surrogate model for granular flows.

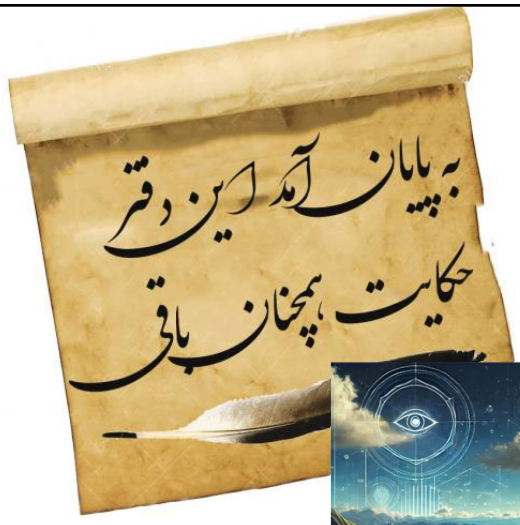
Choi, Y., & Kumar, K. (2024). Computers and Geotechnics, 166, 106015.

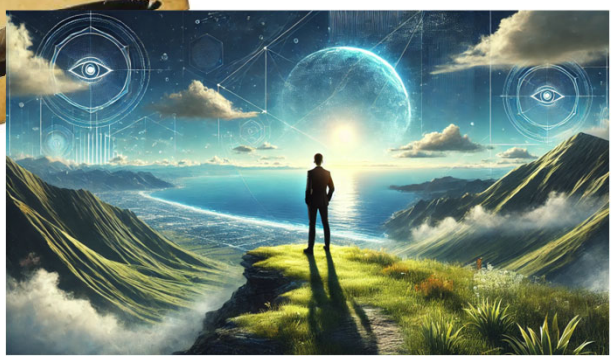
این مقاله از GNNبرای نمایش ذرات به عنوان گره و تعاملات آنها به عنوان لبه استفاده می‌کند. این روش دینامیک فروپاشی دانه‌ای را با دقت پیش‌بینی می‌کند و سرعت محاسباتی آن صدها برابر بیشتر از روش‌های سنتی است.

تمرین

تمرین : یک مساله عمرانی را به روش گراف حل نمایید

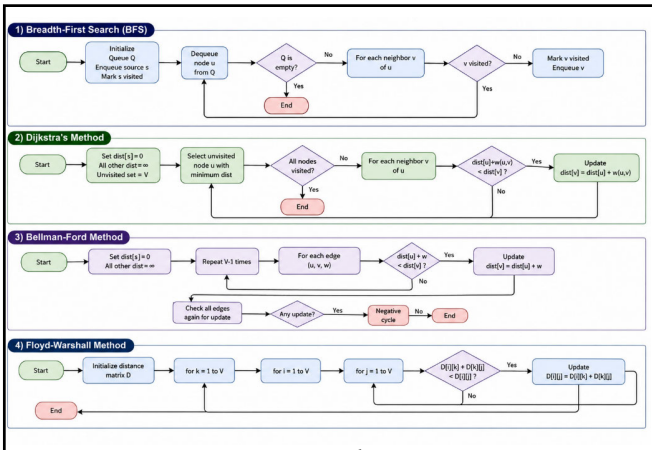
- حداقل ۲۰ گره در نظر بگیرید
- . گره ها از یک تا ۵ حداقل یال داشته باشند
- گراف بدون جهت یا جهت دار می تواند باشد





به دنبال آرزوهایم خواهم رفت
مگر نمیرم عهد بسته ام قبل از

Flowchart



الگوریتم	کاربرد	پیچیدگی
BFS	گراف بدون وزن	O(V+E)
Dijkstra	وزن‌های مثبت	O(E(log V))
Bellman-Ford	وزن‌های منفی	O(VE)
Floyd-Warshall	همه جفت رأس‌ها	O(V^3)