Discrete Mathematics

Contents

1 Recurrence Problems

A recurrence relation is an equation that recursively defines a sequence or multidimensional array of values, once one or more initial terms are given.

1.1 Ordinary Generating Functions (OGF)

Definition 1.1 (Ordinary Generating Function)**.** *The ordinary generating function (OGF) for a sequence $a_0, a_1, a_2, \ldots$ is the formal power series:*

$$G(x) = \sum_{n=0}^{\infty} a_n x^n$$

Pseudo-algorithm: Solving Recurrences with OGFs

1: Let the recurrence be for a_n, valid for $n \geq k$ (for some k).

2: Define the OGF $G(x) = \sum_{n=0}^{\infty} a_n x^n$.

3: Multiply the entire recurrence relation by x^n.

4: Sum the relation from $n = k$ to ∞.

5: Manipulate each term of the sum to express it in terms of $G(x)$ and the initial conditions $(a_0, \ldots, a_{k-1})$.

- $\sum_{n=k}^{\infty} a_n x^n = G(x) - \sum_{i=0}^{k-1} a_i x^i$
- $\sum_{n=k}^{\infty} a_{n-1} x^n = x \sum_{n=k}^{\infty} a_{n-1} x^{n-1} = x \sum_{j=k-1}^{\infty} a_j x^j = x \left(G(x) - \sum_{i=0}^{k-2} a_i x^i \right)$
- $\sum_{n=k}^{\infty} c x^n = c \frac{x^k}{1-x}$ (for a constant c)

6: Solve the resulting algebraic equation for $G(x)$.

7: Decompose $G(x)$ into simpler fractions (using partial fraction decomposition) whose series expansions are known. Common forms:

- $\frac{A}{1-rx} = A \sum_{n=0}^{\infty} (rx)^n = \sum_{n=0}^{\infty} (Ar^n) x^n$
- $\frac{A}{(1-rx)^k} = A \sum_{n=0}^{\infty} \binom{n+k-1}{k-1} (rx)^n$

8: The solution a_n is the coefficient of x^n in the expansion of $G(x)$, denoted $[x^n]G(x)$.

Problem 1.1. *Solve $a_n = 3a_{n-1} + 2$ for $n \geq 1$, with $a_0 = 1$.*

Solution 1.

$$G(x) = \sum_{n=0}^{\infty} a_n x^n = a_0 + \sum_{n=1}^{\infty} a_n x^n$$

$$G(x) = 1 + \sum_{n=1}^{\infty} (3a_{n-1} + 2) x^n$$

$$G(x) = 1 + 3 \sum_{n=1}^{\infty} a_{n-1} x^n + 2 \sum_{n=1}^{\infty} x^n$$

$$G(x) = 1 + 3x \sum_{n=1}^{\infty} a_{n-1} x^{n-1} + 2 \left(\frac{1}{1-x} - 1 \right)$$

$$G(x) = 1 + 3x \sum_{j=0}^{\infty} a_j x^j + \frac{2x}{1-x}$$

$$G(x) = 1 + 3xG(x) + \frac{2x}{1-x}$$

$$G(x)(1-3x) = 1 + \frac{2x}{1-x} = \frac{1-x+2x}{1-x} = \frac{1+x}{1-x}$$

$$G(x) = \frac{1+x}{(1-x)(1-3x)}$$

Using partial fractions: $\frac{1+x}{(1-x)(1-3x)} = \frac{A}{1-x} + \frac{B}{1-3x}$. $1 + x = A(1-3x) + B(1-x)$.

- *Let* $x = 1$*:* $2 = A(1-3) \implies 2 = -2A \implies A = -1$.
- *Let* $x = 1/3$*:* $4/3 = B(1-1/3) \implies 4/3 = B(2/3) \implies B = 2$.

So, $G(x) = \frac{-1}{1-x} + \frac{2}{1-3x}$. *We expand this back into a series:*

$$G(x) = -1\sum_{n=0}^{\infty} x^n + 2\sum_{n=0}^{\infty}(3x)^n = \sum_{n=0}^{\infty}(-1 + 2\cdot 3^n)x^n$$

Therefore, the solution is $a_n = 2\cdot 3^n - 1$.

Problem 1.2. *Solve* $a_n = a_{n-1} + 2a_{n-2}$ *for* $n \geq 2$*, with* $a_0 = 2, a_1 = 1$.

Solution 2.

$$\begin{aligned}
G(x) &= \sum_{n=0}^{\infty} a_n x^n = a_0 + a_1 x + \sum_{n=2}^{\infty} a_n x^n \\
G(x) &= 2 + x + \sum_{n=2}^{\infty}(a_{n-1} + 2a_{n-2})x^n \\
G(x) &= 2 + x + \sum_{n=2}^{\infty} a_{n-1}x^n + 2\sum_{n=2}^{\infty} a_{n-2}x^n \\
G(x) &= 2 + x + x\sum_{n=2}^{\infty} a_{n-1}x^{n-1} + 2x^2\sum_{n=2}^{\infty} a_{n-2}x^{n-2} \\
G(x) &= 2 + x + x\sum_{j=1}^{\infty} a_j x^j + 2x^2\sum_{k=0}^{\infty} a_k x^k \\
G(x) &= 2 + x + x(G(x) - a_0) + 2x^2 G(x) \\
G(x) &= 2 + x + x(G(x) - 2) + 2x^2 G(x) \\
G(x) &= 2 + x + xG(x) - 2x + 2x^2 G(x) \\
G(x)(1 - x - 2x^2) &= 2 - x \\
G(x) &= \frac{2-x}{1-x-2x^2} = \frac{2-x}{(1-2x)(1+x)}
\end{aligned}$$

Using partial fractions: $\frac{2-x}{(1-2x)(1+x)} = \frac{A}{1-2x} + \frac{B}{1+x}$. $2 - x = A(1+x) + B(1-2x)$.

- *Let* $x = -1$*:* $3 = B(1-(-2)) \implies 3 = 3B \implies B = 1$.
- *Let* $x = 1/2$*:* $3/2 = A(1+1/2) \implies 3/2 = A(3/2) \implies A = 1$.

So, $G(x) = \frac{1}{1-2x} + \frac{1}{1+x} = \frac{1}{1-2x} + \frac{1}{1-(-1)x}$.

$$G(x) = \sum_{n=0}^{\infty}(2x)^n + \sum_{n=0}^{\infty}(-1x)^n = \sum_{n=0}^{\infty}(2^n + (-1)^n)x^n$$

Therefore, the solution is $a_n = 2^n + (-1)^n$.

1.2 Population/Bank Balance Problems (Linear First-Order)

These problems model relations of the form $a_n = ra_{n-1} + d$, where r is a rate (e.g., $1 + \text{interest}$) and d is a constant deposit/withdrawal or population change.

Pseudo-algorithm: Solving $a_n = ra_{n-1} + d$

This is a linear non-homogeneous recurrence relation.

Find the homogeneous solution $a_n^{(h)}$:
Solve $a_n = ra_{n-1}$. The characteristic equation is $k - r = 0 \implies k = r$.
The homogeneous solution is $a_n^{(h)} = A \cdot r^n$ for some constant A.
Find the particular solution $a_n^{(p)}$:
Case 1: $r \neq 1$. Guess a constant solution $a_n^{(p)} = B$.
Substitute into the recurrence: $B = rB + d \implies B(1-r) = d \implies B = \frac{d}{1-r}$.
So, $a_n^{(p)} = \frac{d}{1-r}$.
Case 2: $r = 1$. Guess $a_n^{(p)} = Bn$.
Substitute into the recurrence: $Bn = B(n-1) + d \implies Bn = Bn - B + d \implies B = d$.
So, $a_n^{(p)} = dn$.
Find the general solution a_n:
$a_n = a_n^{(h)} + a_n^{(p)}$.
Case 1 ($r \neq 1$): $a_n = Ar^n + \frac{d}{1-r}$.
Case 2 ($r = 1$): $a_n = A \cdot 1^n + dn = A + dn$.
Find the constant A:
Use the initial condition a_0.
Case 1: $a_0 = A \cdot r^0 + \frac{d}{1-r} \implies A = a_0 - \frac{d}{1-r}$.
Case 2: $a_0 = A + d(0) \implies A = a_0$.
Substitute A back into the general solution.

Problem 1.3 (Bank Balance)**.** *You deposit \$5000 into a bank account with 8% annual interest. At the end of each year, you withdraw \$100. Find a formula for the balance a_n after n years.*

Solution 3. *The initial balance is $a_0 = 5000$. The balance after n years is the balance from the previous year, plus 8% interest, minus \$100. $a_n = a_{n-1} + 0.08a_{n-1} - 100 = 1.08a_{n-1} - 100$. This is $a_n = ra_{n-1} + d$ with $r = 1.08$ and $d = -100$. This is Case 1 ($r \neq 1$).*

1. ***Homogeneous solution****:* $a_n^{(h)} = A \cdot (1.08)^n$.
2. ***Particular solution****:* $a_n^{(p)} = \frac{d}{1-r} = \frac{-100}{1-1.08} = \frac{-100}{-0.08} = 1250$.
3. ***General solution****:* $a_n = a_n^{(h)} + a_n^{(p)} = A(1.08)^n + 1250$.
4. ***Find A****: Use $a_0 = 5000$.* $a_0 = A(1.08)^0 + 1250 \implies 5000 = A + 1250 \implies A = 3750$.

The formula is $a_n = 3750(1.08)^n + 1250$.

Problem 1.4 (Population Increase)**.** *An infinitely large pond starts with 200 fish. The fish population grows by 30% each year, but 40 fish are removed by predators. Find a formula for the fish population a_n after n years.*

Solution 4. *The initial population is $a_0 = 200$. The relation is $a_n = a_{n-1} + 0.30a_{n-1} - 40 = 1.30a_{n-1} - 40$. This is $a_n = ra_{n-1} + d$ with $r = 1.30$ and $d = -40$. This is Case 1 ($r \neq 1$).*

1. ***Homogeneous solution****:* $a_n^{(h)} = A \cdot (1.30)^n$.
2. ***Particular solution****:* $a_n^{(p)} = \frac{d}{1-r} = \frac{-40}{1-1.30} = \frac{-40}{-0.30} = \frac{400}{3}$.
3. ***General solution****:* $a_n = a_n^{(h)} + a_n^{(p)} = A(1.30)^n + \frac{400}{3}$.
4. ***Find A****: Use $a_0 = 200$.* $a_0 = A(1.30)^0 + \frac{400}{3} \implies 200 = A + \frac{400}{3} \implies A = \frac{600}{3} - \frac{400}{3} = \frac{200}{3}$.

The formula is $a_n = \frac{200}{3}(1.30)^n + \frac{400}{3}$.

2 Graph Theory

2.1 Basic Definitions

Definition 2.1. *A* ***graph*** $G = (V, E)$ *consists of a set of* **vertices** V *(or nodes) and a set of* **edges** E *(or arcs), where each edge connects two vertices.*

- ***Undirected Graph****: Edges have no orientation.* $e = \{u, v\}$.
- ***Directed Graph (Digraph)****: Edges have orientation.* $e = (u, v)$ *(from* u *to* v*).*
- ***Degree of a Vertex*** $\deg(v)$*: In an undirected graph, the number of edges incident to* v*. (Loops are counted twice).*
- ***In-degree*** $\deg^-(v)$ */* ***Out-degree*** $\deg^+(v)$*: In a digraph, the number of edges entering / leaving* v*.*
- ***Path****: A sequence of vertices* $v_0, v_1, \ldots, v_k$ *where* $\{v_{i-1}, v_i\} \in E$ *(or* $(v_{i-1}, v_i) \in E$*) for all* $i = 1, \ldots, k$*.*
- ***Cycle****: A path that starts and ends at the same vertex* ($v_0 = v_k$) *and does not repeat edges or intermediate vertices.*
- ***Connected Graph****: An undirected graph where there is a path between any two distinct vertices.*

Theorem 2.2 (Handshaking Lemma)**.** *For any undirected graph* $G = (V, E)$*, the sum of the degrees of all vertices is equal to twice the number of edges.*

$$\sum_{v \in V} \deg(v) = 2|E|$$

A corollary is that the number of vertices with odd degree must be even.

Problem 2.1. *Consider* K_4*, the* **complete graph** *on 4 vertices (where every vertex is connected to every other vertex).*

1. *Draw the graph.*
2. *Find the degree of each vertex.*
3. *Verify the Handshaking Lemma.*

Solution 5. 1. ***Drawing****:*

Image Placeholder

A square with its two diagonals drawn. Vertices labeled v_1, v_2, v_3, v_4*.*

Figure 1: A drawing of K_4.

2. ***Degrees****: Let* $V = \{v_1, v_2, v_3, v_4\}$*. Each vertex is connected to the 3 other vertices. So,* $\deg(v_1) = 3$*,* $\deg(v_2) = 3$*,* $\deg(v_3) = 3$*,* $\deg(v_4) = 3$*.*

3. ***Verification**: The total number of edges is $|E| = 6$. Sum of degrees: $\sum \deg(v) = 3+3+3+3 = 12$. Handshaking Lemma: $2|E| = 2 \times 6 = 12$. The lemma holds: $12 = 12$.*

Problem 2.2. *A graph G has 10 vertices. 3 vertices have degree 4, and the other 7 vertices all have the same degree k.*

1. *Find an expression for the number of edges $|E|$ in terms of k.*
2. *What are the possible values for k?*

Solution 6. 1. *By the Handshaking Lemma, $2|E| = \sum \deg(v)$. $2|E| = (3 \times 4) + (7 \times k) = 12 + 7k$. So, $|E| = \frac{12+7k}{2}$.*

2. *Since $|E|$ must be an integer, $12 + 7k$ must be even. Since 12 is even, $7k$ must be even. Since 7 is odd, k must be an even integer. Also, the degree k cannot be greater than $n - 1 = 9$ (in a simple graph). So, the possible values for k are $\{0, 2, 4, 6, 8\}$.*

2.2 Eulerian Circuits and Paths

Definition 2.3. • *An **Eulerian path** is a path in a graph that visits every edge exactly once.*

- *An **Eulerian circuit** (or cycle) is an Eulerian path that starts and ends at the same vertex.*

Theorem 2.4 (Euler's Theorem on Circuits)**.** *A connected graph G has an Eulerian circuit if and only if every vertex of G has an even degree.*

Theorem 2.5 (Euler's Theorem on Paths)**.** *A connected graph G has an Eulerian path if and only if it has exactly zero or two vertices of odd degree.*

- *If zero odd-degree vertices, any Eulerian path is a circuit (starts and ends at same vertex).*
- *If two odd-degree vertices, any Eulerian path must start at one of the odd-degree vertices and end at the other.*

Pseudo-algorithm: Fleury's Algorithm (Conceptual)

This algorithm finds an Eulerian circuit/path.

Check the degrees of all vertices to ensure a circuit (0 odd) or path (2 odd) exists.
Choose a starting vertex:
  If circuit: Start at any vertex v_0.
  If path: Start at one of the two odd-degree vertices, v_0.
Initialize path $P = (v_0)$ and current vertex $u = v_0$.
While there are still unused edges:
  Choose an edge $e = \{u, v\}$ incident to u.
  Crucial Rule: Do not traverse e if it is a **bridge** of the remaining graph (i.e., its removal would disconnect the graph of remaining edges), *unless* there is no other choice (i.e., e is the only edge left at u).
  Add v to P, set $u = v$, and remove e from the graph.
Return P.

Problem 2.3. *The famous "Seven Bridges of Königsberg" problem can be modeled as a multigraph with 4 vertices (land masses) and 7 edges (bridges). Let the vertices be A, B, C, D. The edges are: $\{A,B\}$, $\{A,B\}$, $\{A,C\}$, $\{A,C\}$, $\{A,D\}$, $\{B,D\}$, $\{C,D\}$. Does this graph have an Eulerian circuit or path?*

Solution 7. 1. *Find the degrees of the vertices:*

2. $\deg(A) = 5$ *(to B, B, C, C, D)*

3. $\deg(B) = 3$ *(to A, A, D)*
4. $\deg(C) = 3$ *(to A, A, D)*
5. $\deg(D) = 3$ *(to A, B, C)*
6. *The graph has* ***four*** *vertices of odd degree (5, 3, 3, 3).*
7. *Since the number of odd-degree vertices is not 0 or 2, by Euler's theorems, the graph has* ***neither*** *an Eulerian circuit* ***nor*** *an Eulerian path. It is impossible to cross all seven bridges exactly once.*

Problem 2.4. *Consider a graph with* $V = \{A, B, C, D, E, F\}$ *and edges* $E = \{\{A, B\}, \{A, C\}, \{B, C\}, \{C, D\}, \{C,$ *Does it have an Eulerian circuit or path?*

Solution 8.

1. *Find the degree of each vertex:*
2. $\deg(A) = 2$ *(to B, C)*
3. $\deg(B) = 2$ *(to A, C)*
4. $\deg(C) = 4$ *(to A, B, D, E)*
5. $\deg(D) = 3$ *(to C, E, F)*
6. $\deg(E) = 3$ *(to C, D, F)*
7. $\deg(F) = 2$ *(to D, E)*
8. *The graph has exactly* ***two*** *vertices of odd degree: D and E.*
9. *By Euler's theorem, the graph does not have an Eulerian circuit, but it* ***does have an Eulerian path****.*
10. *Any such path must start at D and end at E, or start at E and end at D.*

2.3 Hamiltonian Circuits and Paths

Definition 2.6.

- *A* ***Hamiltonian path*** *is a path in a graph that visits every vertex exactly once.*
- *A* ***Hamiltonian circuit*** *(or cycle) is a Hamiltonian path that starts and ends at the same vertex (by adding an edge from the last vertex back to the first).*

Note: Unlike Eulerian circuits, there is no simple, necessary and sufficient condition for Hamiltonian circuits. Determining if one exists is a hard problem (NP-complete).

Theorem 2.7 (Dirac's Theorem)**.** *If G is a simple graph with n vertices ($n \geq 3$) such that the degree of every vertex v satisfies* $\deg(v) \geq n/2$*, then G has a Hamiltonian circuit.*

Theorem 2.8 (Ore's Theorem)**.** *If G is a simple graph with n vertices ($n \geq 3$) such that for every pair of non-adjacent vertices u and v,* $\deg(u) + \deg(v) \geq n$*, then G has a Hamiltonian circuit.*

(Note: Dirac's Theorem is a corollary of Ore's Theorem.)

Problem 2.5. *Show that K_5, the complete graph on 5 vertices, has a Hamiltonian circuit.*

Solution 9. *We can show this in two ways:*

1. ***By Construction****: $n = 5$. Let $V = \{1, 2, 3, 4, 5\}$. Since it is a complete graph, every edge exists. A path that visits every vertex is $(1, 2, 3, 4, 5)$. Since the edge $\{5, 1\}$ also exists, we can form the circuit $(1, 2, 3, 4, 5, 1)$. This is a Hamiltonian circuit.*
2. ***By Dirac's Theorem****:*
3. *$n = 5$. In K_5, every vertex is connected to all $n - 1 = 4$ other vertices.*

4. *So, for every vertex* v, $\deg(v) = 4$.
5. *The condition is* $\deg(v) \geq n/2$.
6. $4 \geq 5/2 \implies 4 \geq 2.5$. *This is true.*
7. *Since* $n = 5 \geq 3$ *and all vertices satisfy Dirac's condition,* K_5 *must have a Hamiltonian circuit.*

Problem 2.6. *Consider the "Wheel Graph"* W_5*, which is a* C_4 *cycle (4 vertices on the "rim") plus one central vertex connected to all rim vertices.* $V = \{C, v_1, v_2, v_3, v_4\}$ *and* $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_1\}\} \cup \{\{C, v_1\}, \{C, v_2\}, \{C, v_3\}, \{C, v_4\}\}$. *Does it have a Hamiltonian circuit?*

Solution 10.

1. *The graph has* $n = 5$ *vertices.*
2. *Let's check degrees:*
3. $\deg(C) = 4$ *(connected to all rim vertices).*
4. $\deg(v_1) = \deg(v_2) = \deg(v_3) = \deg(v_4) = 3$ *(connected to 2 rim neighbors and the center* C*).*
5. ***Check Dirac's Theorem****:* $n = 5$, *so* $n/2 = 2.5$.
6. *All vertices have degree* ≥ 3. *Since* $3 \geq 2.5$, *the condition* $\deg(v) \geq n/2$ *is met for all* v.
7. *By Dirac's Theorem,* W_5 ***must have a Hamiltonian circuit****.*
8. ***Example Circuit****:* $(C, v_1, v_2, v_3, v_4, C)$. *This path visits every vertex exactly once and returns to the start.*

3 Algorithms

3.1 Greedy Algorithm (General Concept)

A greedy algorithm builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. It makes a *locally optimal choice* at each step with the hope of finding a *globally optimal solution.* This strategy does not always work, but is very efficient when it does.

Pseudo-algorithm: General Greedy Strategy

Initialize an empty solution set $S = \emptyset$.
Let C be the set of all possible candidates (e.g., coins, edges).
While S is not a complete solution and C is not empty:
    $x = \text{SELECT}(C)$ (Choose the "best" remaining candidate based on a greedy criterion).
    $C = C \setminus \{x\}$ (Remove x from candidates).
    If IS_FEASIBLE($S \cup \{x\}$) (Check if adding x violates any constraints):
        $S = S \cup \{x\}$ (Add x to the solution).
Return S.

Problem 3.1 (Coin Change - Greedy Works)**.** *Find the minimum number of coins to make 68 cents using the standard US coin set: {Quarters (25), Dimes (10), Nickels (5), Pennies (1)}.*

Solution 11. *The greedy strategy is to always take the largest denomination coin possible.*

1. *Amount = 68. Take Quarter (25). Amount = 68 - 25 = 43. Solution: {Q}.*
2. *Amount = 43. Take Quarter (25). Amount = 43 - 25 = 18. Solution: {Q, Q}.*
3. *Amount = 18. Cannot take Quarter. Take Dime (10). Amount = 18 - 10 = 8. Solution: {Q, Q, D}.*
4. *Amount = 8. Cannot take Dime. Take Nickel (5). Amount = 8 - 5 = 3. Solution: {Q, Q, D, N}.*
5. *Amount = 3. Cannot take Nickel. Take Penny (1). Amount = 3 - 1 = 2. Solution: {Q, Q, D, N, P}.*
6. *Amount = 2. Take Penny (1). Amount = 2 - 1 = 1. Solution: {Q, Q, D, N, P, P}.*
7. *Amount = 1. Take Penny (1). Amount = 1 - 1 = 0. Solution: {Q, Q, D, N, P, P, P}.*

The greedy solution is 2 Quarters, 1 Dime, 1 Nickel, 3 Pennies (7 coins). For the standard US coin set, the greedy algorithm is proven to always be optimal.

Problem 3.2 (Coin Change - Greedy Fails)**.** *Find the minimum number of coins to make 30 cents using a custom coin set: {25, 10, 1}.*

Solution 12. *1.* ***Greedy Solution****:*

2. *Amount = 30. Take 25-cent coin. Amount = 30 - 25 = 5. Solution: {25}.*
3. *Amount = 5. Cannot take 10. Take 1-cent coin. Amount = 4. Solution: {25, 1}.*
4. *Amount = 4. Take 1-cent coin. Amount = 3. Solution: {25, 1, 1}.*
5. *Amount = 3. Take 1-cent coin. Amount = 2. Solution: {25, 1, 1, 1}.*
6. *Amount = 2. Take 1-cent coin. Amount = 1. Solution: {25, 1, 1, 1, 1}.*
7. *Amount = 1. Take 1-cent coin. Amount = 0. Solution: {25, 1, 1, 1, 1, 1}.*

8. *The greedy solution uses **6 coins**.*
9. ***Optimal Solution:***
10. *Take 10-cent coin. Amount = 20. Solution: {10}.*
11. *Take 10-cent coin. Amount = 10. Solution: {10, 10}.*
12. *Take 10-cent coin. Amount = 0. Solution: {10, 10, 10}.*
13. *The optimal solution uses **3 coins**.*

In this case, the greedy strategy fails to produce the optimal result.

3.2 Dijkstra's Shortest Path Algorithm

Finds the shortest path from a single source vertex to all other vertices in a weighted graph **with non-negative edge weights**.

Pseudo-algorithm: Dijkstra's Algorithm

```
Input: Graph G, source vertex s.
Initialization:
Create distance array dist[|V|].
Set dist[s] = 0.
Set dist[v] = ∞ for all other v ∈ V \ {s}.
Create a priority queue Q containing all vertices, prioritized by dist.
Create predecessor array prev[|V|], all set to null.
Main Loop:
While Q is not empty:
    u = Q.extract_min() (Get vertex with smallest dist).
    For each neighbor v of u:
        alt_dist = dist[u] + weight(u, v).
        If alt_dist < dist[v]:
            dist[v] = alt_dist (Update distance).
            prev[v] = u (Set predecessor).
            Q.decrease_priority(v, alt_dist) (Update v in priority queue).
Return dist, prev.
```

Problem 3.3. *Find the shortest path from A to all other vertices in the graph: Edges: A-B(4), A-C(2), B-C(5), B-D(10), C-D(3), C-E(8), D-E(4), D-F(1), E-F(6).*

Solution 13. *We trace the dist array.* $dist = \{A: 0, B: \infty, C: \infty, D: \infty, E: \infty, F: \infty\}$. $Q = \{A(0), B(\infty), C(\infty), D(\infty), E(\infty), F(\infty)\}$.

1. ***Visit A (dist=0):***
2. *Neighbors B, C.*
3. $dist[B] = 0 + 4 = 4$. $prev[B] = A$.
4. $dist[C] = 0 + 2 = 2$. $prev[C] = A$.
5. $Q = \{C(2), B(4), D(\infty), E(\infty), F(\infty)\}$.
6. ***Visit C (dist=2):***
7. *Neighbors A, B, D, E. (A is already visited).*
8. $dist[B]$: *alt* $= 2 + 5 = 7$. $7 \not< 4$. *No change.*

9. $dist[D] = 2 + 3 = 5$. $prev[D] = C$.
10. $dist[E] = 2 + 8 = 10$. $prev[E] = C$.
11. $Q = \{B(4), D(5), E(10), F(\infty)\}$.
12. ***Visit B (dist=4):***
13. *Neighbors A, C, D. (A, C are visited).*
14. $dist[D]$: *alt* $= 4 + 10 = 14$. $14 \not< 5$. *No change.*
15. $Q = \{D(5), E(10), F(\infty)\}$.
16. ***Visit D (dist=5):***
17. *Neighbors C, B, E, F. (C, B are visited).*
18. $dist[E]$: *alt* $= 5 + 4 = 9$. $9 < 10$. *Update* $dist[E] = 9$, $prev[E] = D$.
19. $dist[F] = 5 + 1 = 6$. $prev[F] = D$.
20. $Q = \{F(6), E(9)\}$.
21. ***Visit F (dist=6):***
22. *Neighbors D, E. (D is visited).*
23. $dist[E]$: *alt* $= 6 + 6 = 12$. $12 \not< 9$. *No change.*
24. $Q = \{E(9)\}$.
25. ***Visit E (dist=9):***
26. *Neighbors C, D, F. (All visited). No changes.*
27. *Q is empty.*

Final Distances from A:

- $A : 0$
- $B : 4$ *(Path: A-B)*
- $C : 2$ *(Path: A-C)*
- $D : 5$ *(Path: A-C-D)*
- $E : 9$ *(Path: A-C-D-E)*
- $F : 6$ *(Path: A-C-D-F)*

Problem 3.4. *Run Dijkstra from source s on the following graph:* $V = \{s, u, v, x, y\}$, *Edges:* $s - u(10), s - x(5), u - v(1), u - x(2), v - y(4), x - u(3), x - v(9), x - y(2), y - s(7), y - v(6)$.

Solution 14. $dist = \{s : 0, u : \infty, v : \infty, x : \infty, y : \infty\}$. $Q = \{s(0), u(\infty), v(\infty), x(\infty), y(\infty)\}$.

1. ***Visit s (dist=0):***
2. $dist[u] = 0 + 10 = 10$. $prev[u] = s$.
3. $dist[x] = 0 + 5 = 5$. $prev[x] = s$.
4. $Q = \{x(5), u(10), v(\infty), y(\infty)\}$.
5. ***Visit x (dist=5):***
6. $dist[u]$: *alt* $= 5 + 3 = 8$. $8 < 10$. *Update* $dist[u] = 8$, $prev[u] = x$.

7. $dist[v]$: $alt = 5 + 9 = 14$. $prev[v] = x$.
8. $dist[y]$: $alt = 5 + 2 = 7$. $prev[y] = x$.
9. $Q = \{y(7), u(8), v(14)\}$.
10. ***Visit y (dist=7):***
11. $dist[s]$: *visited.*
12. $dist[v]$: $alt = 7 + 6 = 13$. $13 < 14$. *Update* $dist[v] = 13$, $prev[v] = y$.
13. $Q = \{u(8), v(13)\}$.
14. ***Visit u (dist=8):***
15. $dist[v]$: $alt = 8 + 1 = 9$. $9 < 13$. *Update* $dist[v] = 9$, $prev[v] = u$.
16. $Q = \{v(9)\}$.
17. ***Visit v (dist=9):***
18. $dist[y]$: $alt = 9 + 4 = 13$. $13 \not< 7$. *No change.*
19. Q *is empty.*

Final Distances from s:

- $s : 0$
- $u : 8$ *(Path: s-x-u)*
- $v : 9$ *(Path: s-x-u-v)*
- $x : 5$ *(Path: s-x)*
- $y : 7$ *(Path: s-x-y)*

3.3 Floyd-Warshall's All-Pairs Shortest Path Algorithm

Finds the shortest path between **all pairs** of vertices. It can handle negative edge weights, but will detect **negative-weight cycles** (if a diagonal element $dist[i][i]$ becomes negative).

Pseudo-algorithm: Floyd-Warshall

Input: Adjacency matrix W of weights ($W[i][i] = 0$, $W[i][j] = \infty$ if no edge).
Let $n = |V|$.
Create $n \times n$ matrix $D^{(0)} = W$.
Main Loop:
For k from 1 to n: (Allowing vertex k as an intermediate vertex)
  For i from 1 to n: (Source vertex)
    For j from 1 to n: (Destination vertex)
      $D^{(k)}[i][j] = \min(D^{(k-1)}[i][j], \quad D^{(k-1)}[i][k] + D^{(k-1)}[k][j])$ (Is the path $i \to j$ shorter than $i \to k \to j$?)
Return $D^{(n)}$.

Problem 3.5. *Run Floyd-Warshall on the 3-vertex graph with weights:* $W = \begin{pmatrix} 0 & 3 & \infty \\ 2 & 0 & -2 \\ \infty & 1 & 0 \end{pmatrix}$

Solution 15. $D^{(0)} = \begin{pmatrix} 0 & 3 & \infty \\ 2 & 0 & -2 \\ \infty & 1 & 0 \end{pmatrix}$

k = 1 *(Allowing paths through vertex 1):*

- $D^{(1)}[2][3] = \min(D^{(0)}[2][3], D^{(0)}[2][1] + D^{(0)}[1][3]) = \min(-2, 2 + \infty) = -2.$
- $D^{(1)}[3][2] = \min(D^{(0)}[3][2], D^{(0)}[3][1] + D^{(0)}[1][2]) = \min(1, \infty + 3) = 1.$
- *(Other paths are not improved)*

$D^{(1)} = \begin{pmatrix} 0 & 3 & \infty \\ 2 & 0 & -2 \\ \infty & 1 & 0 \end{pmatrix}$ *(No change yet)*

k = 2 *(Allowing paths through vertex 2):*

- $D^{(2)}[1][3] = \min(D^{(1)}[1][3], D^{(1)}[1][2] + D^{(1)}[2][3]) = \min(\infty, 3 + (-2)) = \mathbf{1}.$
- $D^{(2)}[3][1] = \min(D^{(1)}[3][1], D^{(1)}[3][2] + D^{(1)}[2][1]) = \min(\infty, 1 + 2) = \mathbf{3}.$

$D^{(2)} = \begin{pmatrix} 0 & 3 & \mathbf{1} \\ 2 & 0 & -2 \\ \mathbf{3} & 1 & 0 \end{pmatrix}$

k = 3 *(Allowing paths through vertex 3):*

- $D^{(3)}[1][2] = \min(D^{(2)}[1][2], D^{(2)}[1][3] + D^{(2)}[3][2]) = \min(3, 1 + 1) = \mathbf{2}.$
- $D^{(3)}[2][1] = \min(D^{(2)}[2][1], D^{(2)}[2][3] + D^{(2)}[3][1]) = \min(2, -2 + 3) = \mathbf{1}.$

$D^{(3)} = \begin{pmatrix} 0 & \mathbf{2} & 1 \\ \mathbf{1} & 0 & -2 \\ 3 & 1 & 0 \end{pmatrix}$

The final matrix of all-pairs shortest paths is $D^{(3)}$.

Problem 3.6. *Run Floyd-Warshall on the 3-vertex graph with weights:* $W = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix}$

Solution 16. $D^{(0)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix}$

k = 1 *(Allowing paths through vertex 1):*

- $D^{(1)}[2][3] = \min(D^{(0)}[2][3], D^{(0)}[2][1] + D^{(0)}[1][3]) = \min(\infty, 3 + 5) = \mathbf{8}.$

$D^{(1)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & \mathbf{8} \\ \infty & 2 & 0 \end{pmatrix}$

k = 2 *(Allowing paths through vertex 2):*

- $D^{(2)}[1][3] = \min(D^{(1)}[1][3], D^{(1)}[1][2] + D^{(1)}[2][3]) = \min(5, 8 + 8) = 5.$ *(No change)*
- $D^{(2)}[3][1] = \min(D^{(1)}[3][1], D^{(1)}[3][2] + D^{(1)}[2][1]) = \min(\infty, 2 + 3) = \mathbf{5}.$

$D^{(2)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \mathbf{5} & 2 & 0 \end{pmatrix}$

k = 3 *(Allowing paths through vertex 3):*

- $D^{(3)}[1][2] = \min(D^{(2)}[1][2], D^{(2)}[1][3] + D^{(2)}[3][2]) = \min(8, 5 + 2) = \mathbf{7}.$

- $D^{(3)}[2][1] = \min(D^{(2)}[2][1], D^{(2)}[2][3] + D^{(2)}[3][1]) = \min(3, 8+5) = 3$. *(No change)*

$$D^{(3)} = \begin{pmatrix} 0 & \mathbf{7} & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$$

The final matrix of all-pairs shortest paths is $D^{(3)}$.

4 Game Theory (Two-Player Zero-Sum)

In a two-player, zero-sum game, two players (Row Player 1, Column Player 2) make a choice simultaneously. The result is a **payoff matrix**, where the entry a_{ij} is the amount Player 2 pays to Player 1.

- Player 1 (Row) wants to **maximize** the payoff.
- Player 2 (Column) wants to **minimize** the payoff.

4.1 Dominance Method

A strategy is **dominated** if there is another strategy that performs as well or better, no matter what the opponent does. We can eliminate dominated strategies to simplify the game.

Definition 4.1.

- *A* ***row*** *R_i is dominated by R_k if $a_{ij} \le a_{kj}$ for all j, and $a_{ij} < a_{kj}$ for at least one j. (P1 will never play R_i because R_k is always better or equal).*
- *A* ***column*** *C_j is dominated by C_k if $a_{ij} \ge a_{ik}$ for all i, and $a_{ij} > a_{ik}$ for at least one i. (P2 will never play C_j because C_k is always better or equal for P2, meaning a smaller or equal payoff to P1).*

Pseudo-algorithm: Iterative Dominance

Start with the full payoff matrix A.
repeat
  Find any row R_i dominated by another row R_k. If found, remove R_i.
  Find any column C_j dominated by another column C_k. If found, remove C_j.
until no dominated rows or columns can be found.
The resulting matrix is the reduced game.

Problem 4.1 (Election Campaigning)**.** *Two candidates (P1, P2) can campaign in one of three cities (A, B, C). The payoff matrix represents the net gain in votes for P1.*

$$A = \begin{pmatrix} 10 & 5 & 12 \\ 8 & 6 & 9 \\ 4 & 3 & 5 \end{pmatrix}$$

Solution 17.

1. ***Check Rows (P1 wants max):***
2. *R_1 vs R_2: $10 > 8, 5 < 6$. No dominance.*
3. *R_1 vs R_3: $10 > 4, 5 > 3, 12 > 5$. $\implies$ R_1 dominates R_3.*
4. *R_2 vs R_3: $8 > 4, 6 > 3, 9 > 5$. $\implies$ R_2 dominates R_3.*
5. *P1 will never play R_3. We remove it.*

$$A' = \begin{pmatrix} 10 & 5 & 12 \\ 8 & 6 & 9 \end{pmatrix}$$

6. ***Check Columns (P2 wants min):***
7. *C_1 vs C_2: $10 > 5, 8 > 6$. $\implies$ C_2 dominates C_1. (P2 prefers C_2).*
8. *C_1 vs C_3: $10 < 12, 8 < 9$. No dominance.*
9. *C_2 vs C_3: $5 < 12, 6 < 9$. $\implies$ C_2 dominates C_3.*

10. *P2 will never play C_1 (prefers C_2) or C_3 (prefers C_2). We remove C_1 and C_3.*

$$A'' = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

11. ***Check Rows (P1 wants max):***
12. *R_1 (payoff 5) is dominated by R_2 (payoff 6).*
13. *We remove R_1.*

$$A''' = \begin{pmatrix} 6 \end{pmatrix}$$

The game reduces to the single value 6. The solution is (P1 plays R_2, P2 plays C_2) with a value of 6.

Problem 4.2. *Reduce the game:* $A = \begin{pmatrix} -2 & 0 & 3 \\ 4 & 1 & 2 \\ 3 & -1 & 5 \end{pmatrix}$

Solution 18.

1. ***Check Rows (P1 wants max):***
2. *R_1 vs R_2: $-2 < 4, 0 < 1, 3 > 2$. No dominance.*
3. *R_1 vs R_3: $-2 < 3, 0 > -1, 3 < 5$. No dominance.*
4. *R_2 vs R_3: $4 > 3, 1 > -1, 2 < 5$. No dominance.*
5. ***Check Columns (P2 wants min):***
6. *C_1 vs C_2: $-2 < 0, 4 > 1, 3 > -1$. No dominance.*
7. *C_1 vs C_3: $-2 < 3, 4 > 2, 3 < 5$. No dominance.*
8. *C_2 vs C_3: $0 < 3, 1 < 2, -1 < 5$. $\implies$ C_2 dominates C_3.*
9. *P2 will never play C_3 (prefers C_2). We remove it.*

$$A' = \begin{pmatrix} -2 & 0 \\ 4 & 1 \\ 3 & -1 \end{pmatrix}$$

10. ***Check Rows (P1 wants max):***
11. *R_2 vs R_1: $4 > -2, 1 > 0$. $\implies$ R_2 dominates R_1. Remove R_1.*
12. *R_2 vs R_3: $4 > 3, 1 > -1$. $\implies$ R_2 dominates R_3. Remove R_3.*

$$A'' = \begin{pmatrix} 4 & 1 \end{pmatrix}$$

13. ***Check Columns (P2 wants min):***
14. *C_1 (payoff 4) is dominated by C_2 (payoff 1). Remove C_1.*

$$A''' = \begin{pmatrix} 1 \end{pmatrix}$$

The solution is (P1 plays R_2, P2 plays C_2) with a value of 1.

4.2 Maximin Method and Saddle Point

A **saddle point** is an entry in the matrix that is simultaneously the minimum of its row and the maximum of its column. If one exists, it is the stable solution (value) of the game.

Definition 4.2.
- ***Maximin (P1's security):*** *Find the minimum value in each row (P1's worst case). The **maximin** is the maximum of these row minimums.*
- ***Minimax (P2's security):*** *Find the maximum value in each column (P2's worst case). The **minimax** is the minimum of these column maximums.*

Theorem 4.3 (Minimax Theorem)**.** *For every two-player zero-sum game, maximin $\leq$ minimax. If maximin $=$ minimax, the game has a saddle point, and this value is the **value of the game**. The optimal strategies are the pure strategies (row, column) that produce this value. If maximin $<$ minimax, no saddle point exists, and the optimal solution involves a mixed strategy.*

Pseudo-algorithm: Finding Saddle Points

1: For each row i, find $m_i = \min_j(a_{ij})$.
2: Calculate maximin $= \max_i(m_i)$.
3: For each column j, find $M_j = \max_i(a_{ij})$.
4: Calculate minimax $= \min_j(M_j)$.
5: If maximin $=$ minimax, a saddle point exists at the (row, column) that yielded this value.
6: Else, no saddle point exists.

Problem 4.3 (Election Campaigning)**.** *Find the saddle point, if one exists.*

$$A = \begin{pmatrix} 4 & 2 & 5 \\ 6 & \mathbf{3} & 7 \\ 1 & 2 & 4 \end{pmatrix}$$

Solution 19.
1. ***Row Mins (P1):***
2. $R_1 : \min(4, 2, 5) = 2$
3. $R_2 : \min(6, 3, 7) = 3$
4. $R_3 : \min(1, 2, 4) = 1$
5. *Maximin* $= \max(2, 3, 1) = \mathbf{3}$
6. ***Column Maxs (P2):***
7. $C_1 : \max(4, 6, 1) = 6$
8. $C_2 : \max(2, 3, 2) = 3$
9. $C_3 : \max(5, 7, 4) = 7$
10. *Minimax* $= \min(6, 3, 7) = \mathbf{3}$

Since maximin $=$ minimax $= 3$, a saddle point exists. It is at $a_{22} = 3$. This is the minimum of its row (Row 2) and the maximum of its column (Column 2). The optimal strategy is: P1 plays R_2, P2 plays C_2. The value of the game is 3.

Problem 4.4. *Find the saddle point, if one exists.*

$$A = \begin{pmatrix} 10 & 2 & 5 \\ 8 & 4 & 7 \\ 9 & 3 & 6 \end{pmatrix}$$

Solution 20. 1. ***Row Mins (P1):***

2. $R_1 : \min(10, 2, 5) = 2$
3. $R_2 : \min(8, 4, 7) = 4$
4. $R_3 : \min(9, 3, 6) = 3$
5. *Maximin* $= \max(2, 4, 3) = \mathbf{4}$
6. ***Column Maxs (P2):***
7. $C_1 : \max(10, 8, 9) = 10$
8. $C_2 : \max(2, 4, 3) = 4$
9. $C_3 : \max(5, 7, 6) = 7$
10. *Minimax* $= \min(10, 4, 7) = \mathbf{4}$

Since maximin = minimax = 4, a saddle point exists. It is at $a_{22} = 4$. The optimal strategy is: P1 plays R_2, P2 plays C_2. The value of the game is 4.

4.3 Mixed Strategy (Graphical Method)

Used for $2 \times N$ or $M \times 2$ games with no saddle point. We will solve a 2×3 game.

Pseudo-algorithm: Graphical Method ($2 \times N$ Game)

Input: $2 \times N$ matrix A.
Let P1 play R_1 with probability p and R_2 with probability $(1 - p)$.
For each of P2's columns C_j $(j = 1, \ldots, N)$, write the expected payoff E_j for P1:
$E_j(p) = a_{1j} \cdot p + a_{2j} \cdot (1 - p)$.
Draw a graph with p on the x-axis (from 0 to 1).
Plot all N lines $E_j(p)$ on this graph.
Identify the **lower envelope** of these lines. This represents P1's minimum guaranteed payoff for any choice of p. (This is the "bottom" boundary of the shaded region).
Find the highest point on this lower envelope. This is the **maximin point**. This point will be the intersection of two of the lines, say E_k and E_l.
Solve $E_k(p) = E_l(p)$ to find the optimal probability p^*.
The value of the game v is $E_k(p^*)$ (or $E_l(p^*)$).
P1's optimal strategy: $(p^*, 1 - p^*)$.
P2's optimal strategy: P2 only uses the strategies C_k and C_l that define the maximin point. Solve the 2×2 subgame $\begin{pmatrix} a_{1k} & a_{1l} \\ a_{2k} & a_{2l} \end{pmatrix}$ to find P2's probabilities (q_k, q_l).

Problem 4.5 (Election Campaigning)**.** *Solve the game (no saddle point exists):*

$$A = \begin{pmatrix} 1 & 8 & 6 \\ 7 & 2 & 4 \end{pmatrix}$$

Solution 21. *Let P1 play R_1 with prob p and R_2 with prob $(1 - p)$. Expected payoffs for P1 against each of P2's choices:*

- *C_1: $E_1(p) = 1p + 7(1 - p) = 7 - 6p$*
- *C_2: $E_2(p) = 8p + 2(1 - p) = 2 + 6p$*
- *C_3: $E_3(p) = 6p + 4(1 - p) = 4 + 2p$*

Graphical Analysis:

- E_1: *Starts at 7 (at* $p = 0$*) and goes down to 1 (at* $p = 1$*).*
- E_2: *Starts at 2 (at* $p = 0$*) and goes up to 8 (at* $p = 1$*).*
- E_3: *Starts at 4 (at* $p = 0$*) and goes up to 6 (at* $p = 1$*).*

The lower envelope is formed by E_1 *(from* $p = 0$ *to an intersection) and* E_2 *(from an intersection to* $p = 1$*). The line* E_3 *is always above the intersection of* E_1 *and* E_2*, so it is irrelevant (dominated in the mix). The maximin point is the intersection of* E_1 *and* E_2*.*

1. ***Find*** p^*: $E_1(p) = E_2(p) \implies 7 - 6p = 2 + 6p \implies 5 = 12p \implies p^* = \mathbf{5/12}$. *P1's strategy: Play* R_1 *with prob 5/12,* R_2 *with prob* $1 - 5/12 = 7/12$*.*
2. ***Find Value*** v: $v = 7 - 6(5/12) = 7 - 5/2 = 4.5$. *(Check:* $v = 2 + 6(5/12) = 2 + 5/2 = 4.5$*).*
3. ***Find P2's strategy****: P2 only uses* C_1 *(prob* q*) and* C_2 *(prob* $1-q$*). We set P1's expected payoff to* v *for both of P2's pure strategies:*
4. *vs* R_1: $1q + 8(1 - q) = 4.5 \implies 8 - 7q = 4.5 \implies 3.5 = 7q \implies q = 0.5$.
5. *vs* R_2: $7q + 2(1 - q) = 4.5 \implies 2 + 5q = 4.5 \implies 5q = 2.5 \implies q = 0.5$. *P2's strategy: Play* C_1 *with prob 1/2,* C_2 *with prob 1/2, and* C_3 *with prob 0.*

Solution*: P1 plays* $(5/12, 7/12)$*, P2 plays* $(1/2, 1/2, 0)$*, Value = 4.5.*

Problem 4.6. *Solve the game:*

$$A = \begin{pmatrix} 9 & 2 \\ 3 & 8 \end{pmatrix}$$

Solution 22. *(Maximin = 3, Minimax = 8. No saddle point). Let P1 play* R_1 *with prob* p *and* R_2 *with prob* $(1 - p)$*.*

- C_1: $E_1(p) = 9p + 3(1 - p) = 3 + 6p$
- C_2: $E_2(p) = 2p + 8(1 - p) = 8 - 6p$

The maximin point is the intersection of the two lines.

1. ***Find*** p^*: $E_1(p) = E_2(p) \implies 3 + 6p = 8 - 6p \implies 12p = 5 \implies p^* = \mathbf{5/12}$. *P1's strategy:* $(5/12, 7/12)$.
2. ***Find Value*** v: $v = 3 + 6(5/12) = 3 + 5/2 = 5.5$.
3. ***Find P2's strategy*** *(prob* q *for* C_1*,* $1 - q$ *for* C_2*):*
4. *vs* R_1: $9q + 2(1 - q) = 5.5 \implies 2 + 7q = 5.5 \implies 7q = 3.5 \implies q = 0.5$. *P2's strategy:* $(1/2, 1/2)$.

Solution*: P1 plays* $(5/12, 7/12)$*, P2 plays* $(1/2, 1/2)$*, Value = 5.5.*