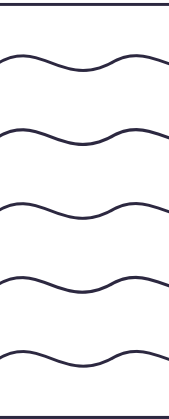
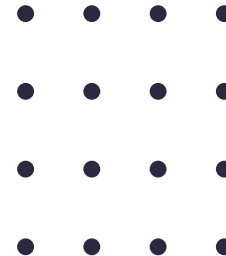
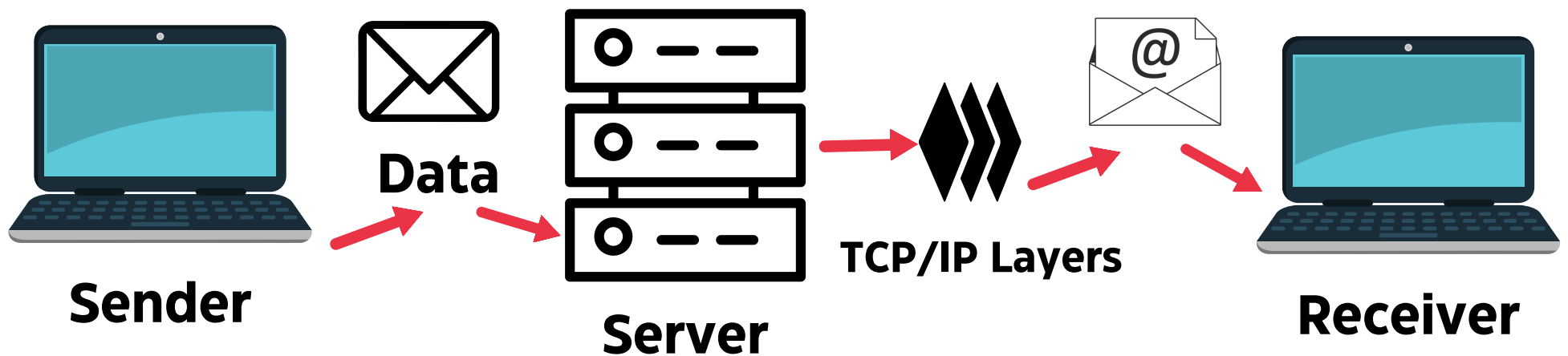


TCP/IP COMMUNICATION



What is **TCP/IP** and what does it do?

TCP/IP (Transmission Control Protocol/Internet Protocol) is a collection of communication protocols that enables the transmission and **exchange of data between computers and other devices** over networks.



To ensure that, each message reaches its final destination accurately, the TCP/IP model divides its data into packets and combines them at the other end, which helps in maintaining the accuracy of the data while transferring from one end to another end.

How packetizing data helps ensure data accuracy?

Efficient transmission: Breaking data into smaller packets allows for **more efficient transmission across the network**. Large amounts of data can be divided into smaller, manageable units, which can be sent separately and reassembled at the destination. This approach reduces the impact of network congestion and ensures a smoother flow of data.

Error detection and correction: When data is divided into packets, **each packet includes additional information, such as error detection codes**. These codes enable the receiving device to verify the integrity of each packet. If errors are detected during transmission, **mechanisms like error detection codes can help identify and discard corrupted packets**. This process allows for retransmission of only the affected packets, minimizing the impact of errors on the overall data accuracy.

Reliable delivery: Packet-based protocols, such as TCP in the TCP/IP suite, provide reliable delivery of data. **Each packet contains a sequence number, allowing the receiving device to reorder the packets** if they arrive out of order.

Basic difference between TCP and IP

TCP

.TCP is a **connection-oriented protocol**.

.It establishes a connection-oriented communication channel and **handles features like error detection, flow control, congestion control**, and retransmission of lost or corrupted packets.

.It is **responsible for ensuring reliable and ordered data delivery** between two devices.

IP

.IP is a **connectionless protocol**.

.It provides the **foundation for data transmission by assigning unique IP addresses to devices**, encapsulating data into packets, and routing them through different networks.

.It is **responsible for addressing, routing**, and fragmenting data packets across networks. It **does not guarantee reliable delivery or order of packets**.

Layers of TCP/IP Model

1) Application Layer

2) Transport Layer

3) Internet Layer

4) Network Access Layer

1) Application Layer

The Application layer is the topmost layer in the TCP/IP model. **It includes the protocols and processes that interact directly with applications and end-users.** This layer includes protocols like HTTP (Hypertext Transfer Protocol) for web browsing, SMTP (Simple Mail Transfer Protocol) for email, FTP (File Transfer Protocol) for file transfer, DNS (Domain Name System) for name resolution, and many others. The Application layer protocols provide services and functionality to support specific applications.

2) Transport Layer

The Transport layer is responsible for end-to-end communication between hosts. It **ensures reliable and ordered delivery of data by dividing data into segments, assigning sequence numbers,** and implementing error detection, flow control, and congestion control mechanisms.

3) Internet Layer

The Internet layer is where the IP (Internet Protocol) resides. It provides the addressing, routing, and fragmentation of data packets across different networks. The Internet layer **handles the encapsulation of data into IP packets, assigns unique IP addresses to devices**, and determines the best path for packet delivery using routing protocols.

4) Network Access Layer

The Network Interface layer is **responsible for the physical transmission of data between devices on the same network**. It encompasses the protocols and hardware technologies that directly interface with the physical network, such as Ethernet, Wi-Fi, and others.

How communication is made between server and bots?

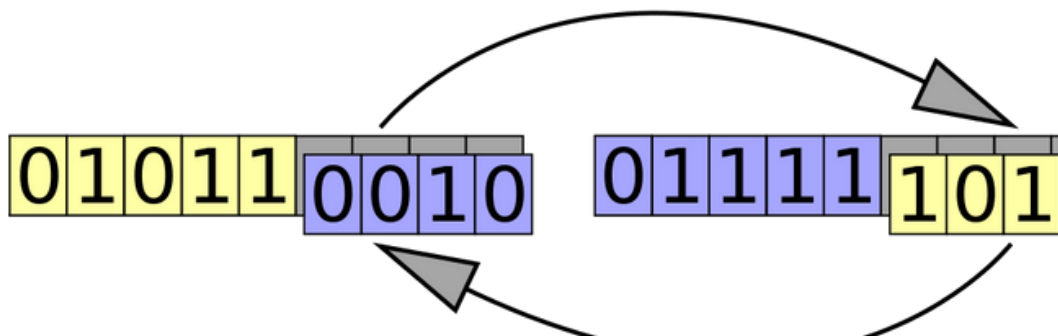
The bot and server communicate by creating a virtual connection, exchanging messages, and confirming that the messages are received. They make sure the data is delivered reliably and in order. When they're done, they agree to close the connection.

- The bot sends a message to the server to start the connection.
- The server receives the bot's message and responds with a confirmation.
- The bot receives the confirmation and acknowledges it.
- Once the connection is established, the bot and server can share information.
- The bot splits the information into smaller parts and sends them to the server.
- The server confirms the receipt of each part.
- If the bot doesn't receive confirmation from the server, it assumes something went wrong and sends that part again.
- When the bot finishes sending data, it lets the server know it wants to close the connection.
- Finally, both sides confirm the termination, and the connection is closed



Genetic Algorithm

0101111010 011111001



What is genetic algorithm?

A genetic algorithm is one of a class of algorithms that **searches a solution space for the optimal solution to a problem.**

Inspired by the process of natural selection, it mimics the principles of evolution, such as mutation, crossover, and selection, to iteratively improve a population of candidate solutions to a problem.

Let's understand it with an example

Let's say the length of the binary string is 4, and the **mathematical function to maximize is the sum of the binary digits.**

So, first of all we will **Create a population** of binary strings of length 4 randomly.

Example:

Individual 1: 1010

Individual 2: 1100

Individual 3: 1111

Individual 4: 1110

Individual 5: 0001

Now we will **evaluate and assign the fitness score** accordingly

Example:

Individual 1: $1 + 0 + 1 + 0 = 2$

Individual 2: $1 + 1 + 0 + 1 = 3$

Individual 3: $1 + 1 + 1 + 0 = 3$

Individual 4: $1 + 1 + 0 + 0 = 2$

Individual 5: $0 + 0 + 0 + 1 = 1$

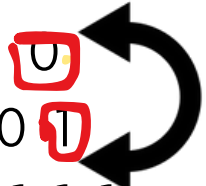
Fitness score:

Individual 3 = Individual 4 > Individual 2 = Individual 1 > Individual 5

Now we will, choose individuals from the population to serve as parents for the next generation based on their fitness. **Higher fitness values increase the chances of being selected**, reflecting the principle of "survival of the fittest." And then'll perform crossover between selected parents to create offspring.

Example (single-point crossover):

Parent 1: 1 1 1 0
Parent 2: 1 1 0 1
Offspring 1: 1 1 1 1
Offspring 2: 1 1 0 0

A diagram illustrating single-point crossover. Two curved arrows originate from the fourth bit of each parent. One arrow points from the '0' in Parent 1 to the fourth position of Offspring 1. The other arrow points from the '1' in Parent 2 to the fourth position of Offspring 2. The crossover point is at the end of the third bit.

Now we will introduce **random changes to the offspring's** genetic material.

Example:

Offspring 1 (no mutation): 1111

Offspring 2 (**mutation**): 1110

After that we'll **replace the old population with the new offspring population.**

If the maximum number of generations is reached or the **target fitness is achieved, we will stop the algorithm.**

Otherwise, we will again go to step 2 that is we will evaluate the present new generation and next step will run accordingly

By repeating these steps iteratively, the genetic algorithm explores the search space, gradually improving the fitness of the population, and potentially finding the optimal solution.

So, here's a **step-by-step breakdown** of how a genetic algorithm typically operates:

Initialization

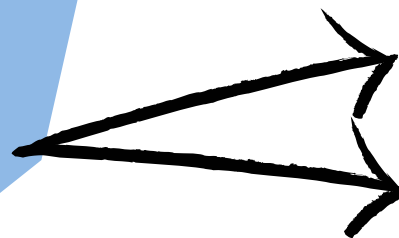
Evaluation

Selection

Genetic Operators

Crossover

Mutation



Replacement

Termination

Travelling Salesman Problem

The Traveling Salesman Problem (TSP) is a classic optimization problem in computer science and mathematics. **It involves travelling a set of cities and return to the starting city, while visiting each city exactly once such that it costs salesman the least amount of money possible**

The problem is considered NP-hard, meaning that as the number of cities increases, the time required to find an optimal solution grows exponentially.

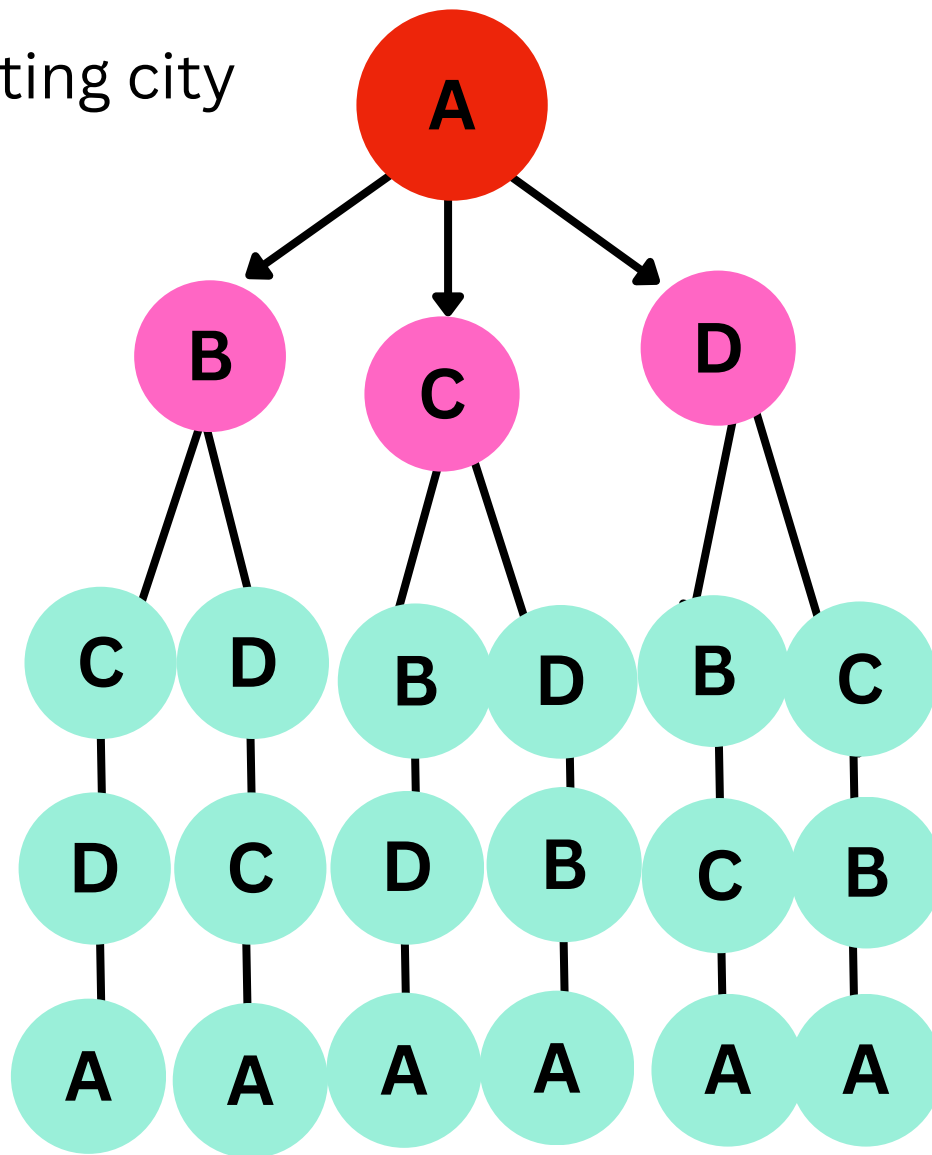
Example:

The cost between each pair of cities are written, forming a **price matrix** such as A to B - \$50, A to C -\$25 and so on.

Now we will generate **all possible permutations of the cities.**

	A	B	C	D
A	0	50	25	75
B	50	0	30	40
C	25	30	0	15
D	75	40	15	0

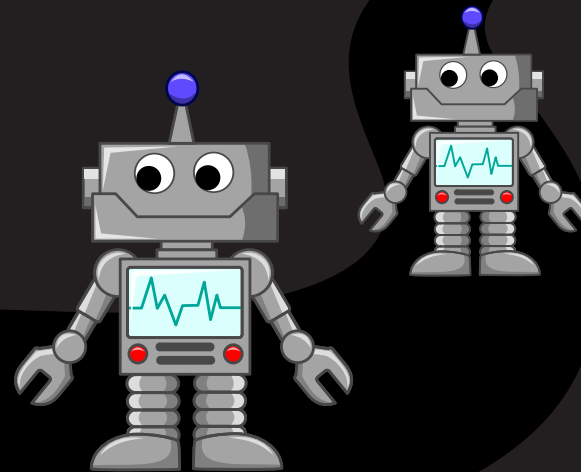
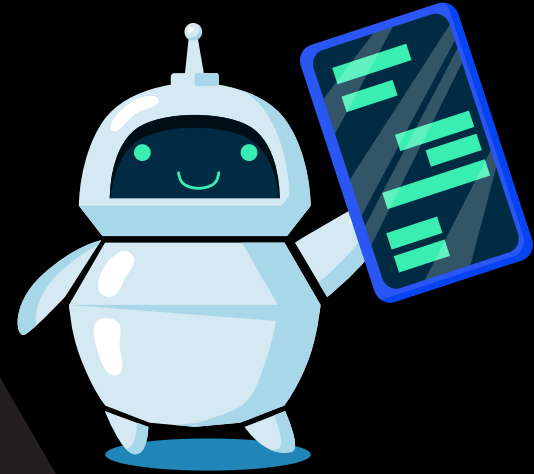
Let **A** be the starting city



Total Cost: 170 130 170 130 170 170

The optimal solution is **ABDCA** or **ACDBA** with a total price of \$**130**.

ROBOT PATH PLANNING



What is Robot Path Planning?

Robot path planning is the process of determining the **optimal path for a robot to move from its current location to a desired goal location**, while avoiding obstacles in its environment.

RRT(Rapidly-exploring Random Trees):
a popular algorithm used in robot path planning

Steps involved in RRT:

Initialization

algorithm starts by placing the initial position of the robot as the root of the tree.

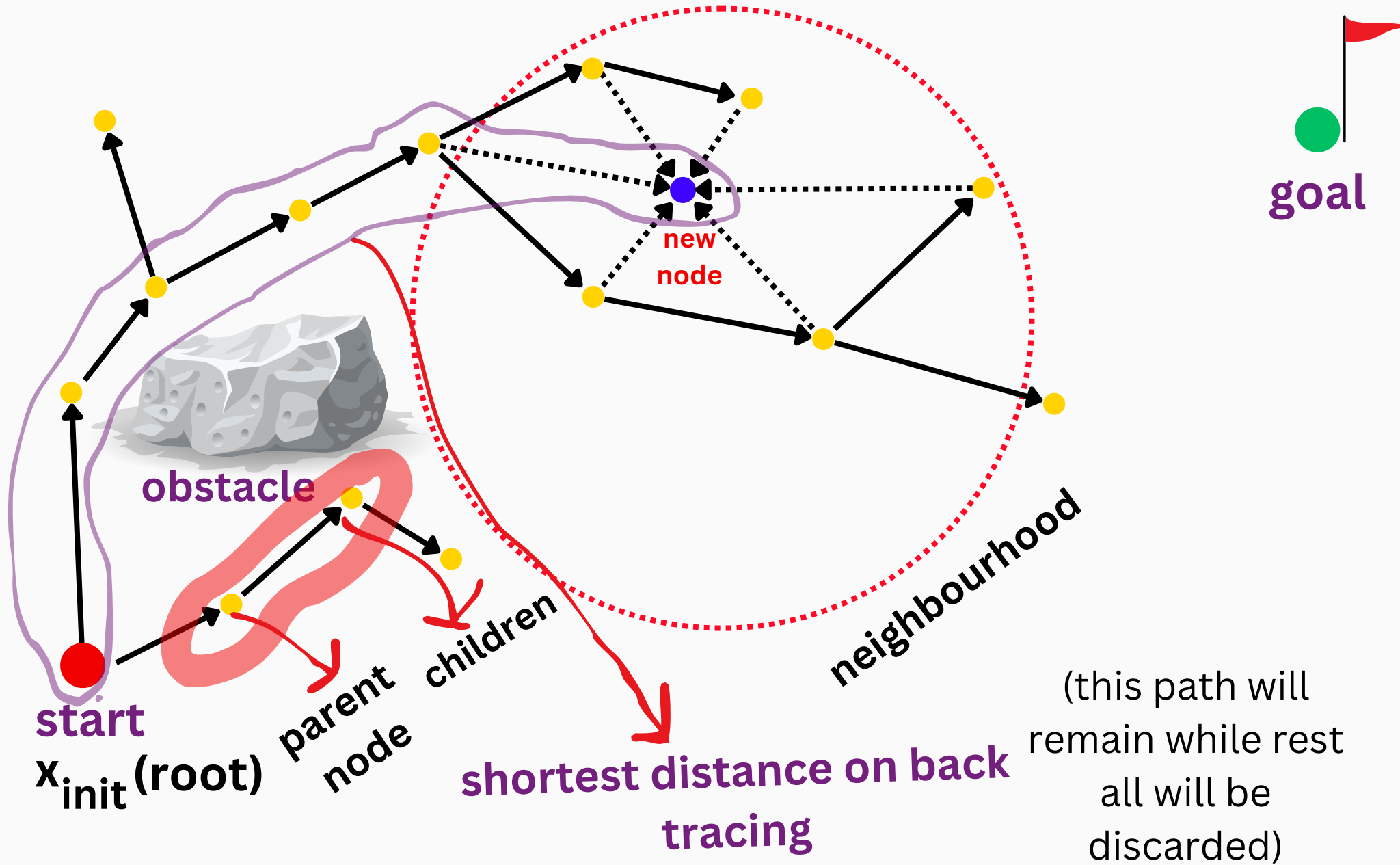
Random sampling

generates random samples in the environment which represents a position for the robot to explore

Nearest neighbor

The algorithm identifies the nearest node in the existing tree to the sampled position. This node becomes the parent of the newly sampled position.

Diagram for RRT:



Motion planning

A new configuration is generated by taking a small step from the parent node towards the sampled position

Collision checking

If the newly generated configuration is in collision, the algorithm discards it and proceeds to the next random sample.

Adding nodes

If the configuration is collision-free, it is added as a new node to the tree.

Goal check

At each iteration, the algorithm checks if the newly added node is close enough to the goal location

Iteration

Steps are repeated until the algorithm finds a feasible path or reaches a maximum number of iterations.

Path extraction

The path from the initial position to the goal can be extracted by tracing back from the goal node to the root node along the connections.

RRT v/s RRT*

RRT is a simpler algorithm that rapidly explores the configuration space and finds feasible paths but does not guarantee optimality.

RRT* improves upon RRT by adding an optimization step, allowing rewiring, and aiming for optimal paths. RRT* is more computationally intensive but provides better guarantees in terms of optimality.