Indices

- Artificial intelligence
- AgentSpeak
- Exercises AgentSpeak
- Algoritmos
- Aritmética
- Algebra
- Algebra 2
- Investigación de operaciones
- Problemas
- MLP
- Ant Colony
- Programación Dinámica
- Tesis
- Micro
- Otros
- Estados de México
- Tablas de multiplicar
- Letras griegas

Artificial intelligence

1 Introduction

We are called

Homosapiens (Hombre sabio) because our **intelligence**

we always try to understand

how we think, how a mere body of matter can perceive, understand, predict and manipulate a world far larger than itself.

Artificial Intelligence

goes further, it attemps not just understand

but also

build intelligent entities

it started

after world war 2 and it was named in 1965

is relevant

to any intellectual task, which makes it a universal field

1.1 ¿What is AI?

We will try to address now what is an AI, we will show 4 definitions

1	Thinking	Is about though process and reasoning
2	Acting	Address behavior
3	Humanly	refers to success in terms of human performance
4	Rational	Is about though process and reasoning

Definitions

	Humanly	Rationally
Thinking	"The exciting new effort to make computers think... machines with minds, in the full and literal sense." (Haugeland, 1985)	"The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting	"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990)	"Computational Intelligence is the study of the design of intelligent agents." (Poole et al., 1998

1 Introduction

We are called

Homosapiens (Hombre sabio) because our

we always try to understand

how we think, how a mere body of matter can perceive, understand, predict and a world far larger than itself.

Artificial Intelligence

goes further, it attemps not just

but also

intelligent entities

it started

after and it was named in 1965

is relevant

to any , which makes it a universal field

1.1 ¿What is AI?

We will try to address now what is an AI, we will show 4 definitions

1		Is about though process and reasoning
2		Address behavior
3		refers to success in terms of human performance
4		Is about though process and reasoning

Definitions

	Humanly	Rationally
Thinking	"The exciting new effort to make computers think... machines with minds, in the full and literal sense." (Haugeland, 1985)	"The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting	"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990)	"Computational Intelligence is the study of the design of intelligent agents." (Poole et al., 1998

1.1.3 Acting humanly: The turing test

A	*In 1950 **Alan turing** propose a **test**, which will prove if a machine is intelligent in the **acting** approach.*
B	it involves an interrogator which ask some questions to the computer, the computer passes the test if the human **cannot** tell if its a person or a computer.

In order to to this

The computer would need the following capabilities:

1	**Natural language processing**	to enable it to **communicate** succesfully in english
2	**Knowledge representation**	to **store** what it knows or hears
3	**Automated reasoning**	to use the stored information to **answer** questions and to draw new conclusions
4	**Machine learning**	to **adapt** to new circumstances and to detect and extrapolate patterns

Total turing test

includes the next capabilities

5	**Computer vision**	to **perceive** objects
6	**Robotics**	to **manipulate** objects and move them

This six disciplines compose the most of AI, however AI research have devoted **little** effort to passing the Turing test

They think doing a **duplicate** is no more important than to study the basic principles.

1.1.4 Acting Rationally: the rational agent approach

Agent

Is something that **acts**, comes from the latin *agere*, which means ***to do***

Rational Agent

Is one that acts, to achieve the **best** outcomes or sometimes the **best expected** outcome

Emphasis

The emphasis of AI is in correct **inferences**, but this is ***part*** of being a rational agent, in some situations, there is probably no **correct thing to do**, but still something must be done

Perfect rationality

Is always **doing the right thing**, which is not **feasible** in complicated environments, like for example having computational demands to high. although is a good point to start for analysis.

Limited Rationality

Acting appropriately when there is not enough **time** to do all computations one might like.

1.1.3 Acting humanly: The turing test

A	*In 1950 propose a , which will prove if a machine is intelligent in the approach.*
B	it involves an interrogator which ask some questions to the computer, the computer passes the test if the human tell if its a person or a computer.

In order to to this

The computer would need the following capabilities:

1	**Natural language processing**	to enable it to succesfully in english
2	**Knowledge representation**	to what it knows or hears
3	**Automated reasoning**	to use the stored information to questions and to draw new conclusions
4	**Machine learning**	to to new circumstances and to detect and extrapolate patterns

Total turing test

includes the next capabilities

5	**Computer vision**	to objects
6	**Robotics**	to objects and move them

This six disciplines compose the most of AI, however AI research have devoted effort to passing the Turing test

They think doing a is no more important than to study the basic principles.

1.1.4 Acting Rationally: the rational agent approach

Agent

Is something that , comes from the latin *agere*, which means

Rational Agent

Is one that acts, to achieve the outcomes or sometimes the outcome

Emphasis

The emphasis of AI is in correct , but this is of being a rational agent, in some situations, there is probably no , but still something must be done

Perfect rationality

Is always , which is not in complicated environments, like for example having computational demands to high. although is a good point to start for analysis.

Limited Rationality

Acting appropriately when there is not enough to do all computations one might like.

1.2 Foundations

*There has been many disciplines that contributed ideas, viewpoints, and techniques to **AI**.*

1 Philosophy

Aristotle was the first to formulate a precise set of laws governing the rational part of the mind. He was a strong member of **Rationalism**, which view is that all can be explained rationally,

Descartes was a proponent of **dualism**, that is there is a part of the human mind (soul or spirit) that is outside of nature.

Materialism holds the view that the brain is the mind.

2 Matemathics

it helps with logic, computations, and probability, mathematical development began with the work of **George Boole**

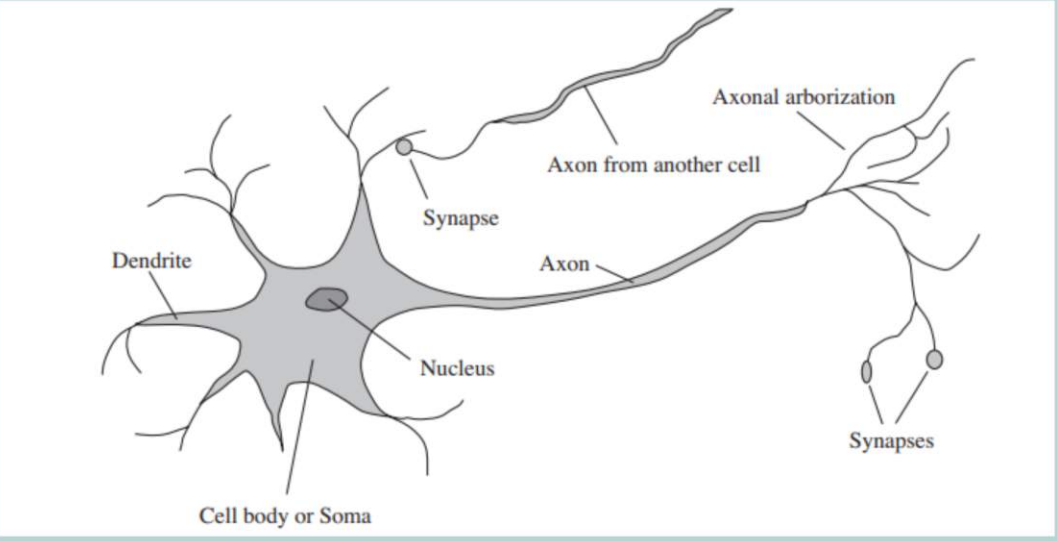
3 Economics

Is about agents maximizing their own **economic** well-being, is about studying how people make choices that lead to preferred outcomes. it began in 1776 with Scottish philosopher **Adam Smith**

4 Neuroscience

Is the study of the nervous system, particularly the brain, the brain is widely recognized as the seat of consciousness, a brain consists of **neurons**, this thanks to Camillo Golgi in 1873

1.1.4 Foundations



Parts of a Neuron

1. Each **neuron** cosists of a **cell body** or **Soma**
2. Each **cell body** have one **Nucleus**
3. branching out from the **cell body** are a number of fibers called **dendrites**
4. there is a single long fiber called the **axon**, that stretches a long distance
5. A **neuron** makes connections with other **neurons** at junctions called **synapses**

Signals are propagated from **neuron** to **neuron**, this signals control brain activity in the short term and also enable long-term changes in the connectivity of **neurons,** computers have a **cycle time** that is million times faster than a brain, but the brain makes up for that with far more **storage** and interconnection

1.2 Foundations

There has been many disciplines that contributed ideas, viewpoints, and techniques to .

1 Philosophy

was the first to formulate a precise set of laws governing the rational part of the mind. He was a strong member of , which view is that all can be explained rationally,

Descartes was a proponent of , that is there is a part of the human mind (soul or spirit) that is outside of nature.

holds the view that the brain is the mind.

2 Matemathics

it helps with logic, computations, and probability, mathematical development began with the work of

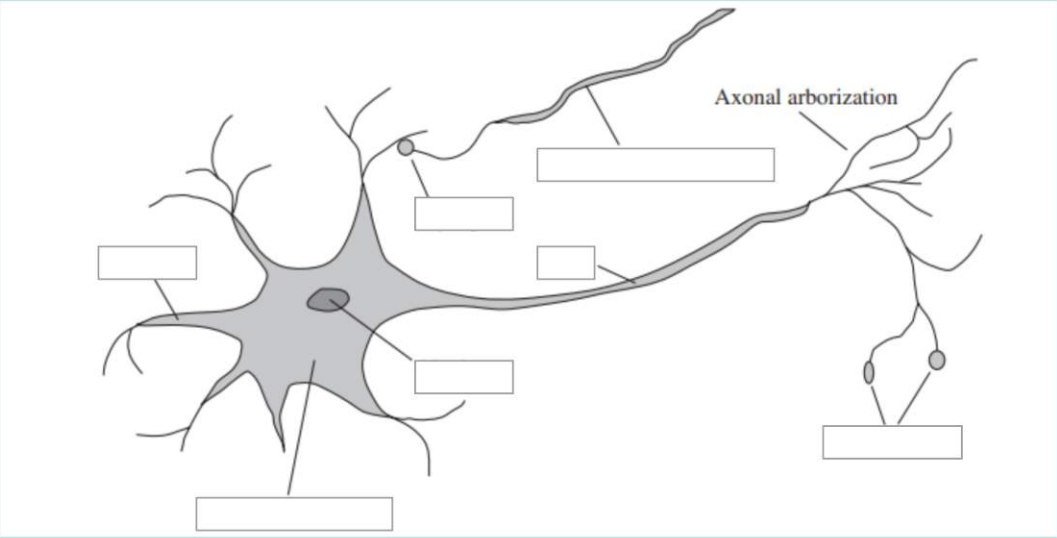
3 Economics

Is about agents maximizing their own well-being, is about studying how people make choices that lead to preferred outcomes. it began in 1776 with Scottish philosopher

4 Neuroscience

Is the study of the nervous system, particularly the brain, the brain is widely recognized as the seat of consciousness, a brain consists of , this thanks to Camillo Golgi in 1873

1.1.4 Foundations



Parts of a Neuron

1. Each **neuron** cosists of a **cell body** or **Soma**
2. Each **cell body** have one **Nucleus**
3. branching out from the **cell body** are a number of fibers called **dendrites**
4. there is a single long fiber called the **axon**, that stretches a long distance
5. A **neuron** makes connections with other **neurons** at junctions called **synapses**

Signals are propagated from **neuron** to **neuron**, this signals control brain activity in the and also enable changes in the connectivity of **neurons,** computers have a that is million times faster than a brain, but the brain makes up for that with far more and interconnection

1.3 Gestation and birth

*The first work recognized as **AI** was done by Warren McCulloch and Walter Pitts in 1943*

They proposed a model of artificial **neurons** in which each neuron is characterized as being **on** or **off**, they show that any function could be computed by some network of neurons.

Birth

in 1951 **John McCarthy**, propose to 10 attendees to study AI, they work on the idea of duplicating human faculties such as creativity, self-improvement, and language use.

Enthusiasm

It was astonishing whenever a computer did anything remotely clever, John McCarthy referred to this period as the "**Look, Ma, no hands!**" era.

Lisp

John McCarthy defined in 1958 the high-level language **Lisp**, which was the dominant AI programming language for 30 years.

Machine evolution

Genetic algorithms, were based on the undoubtedly correct belief that by making an appropriate series of small mutations to a machine-code program, one can generate a program with good performance for any particular task.

1.4 Conclusion

Philosophers

made AI conceivable by considering the ideas that the mind is in some ways like a **machine**

Mathematicians

They set the groundwork for understanding computation and reasoning about **algorithms**.

Economists

formalized the problem of making decisions that maximize the expected **outcome** to the decision maker.

Neuroscientists

discovered some facts about how the **brain** works and the ways in which it is similar to and different from computers

Psychologists

adopted the idea that humans and animals can be considered information processing **machines**

Computer engineers

provided the ever-more-powerful **machines** that make AI applications possible.

1.3 Gestation and birth

The first work recognized as was done by Warren McCulloch and Walter Pitts in 1943

They proposed a model of artificial in which each neuron is characterized as being or , they show that any function could be computed by some network of neurons.

Birth

in 1951 , propose to 10 attendees to study AI, they work on the idea of duplicating human faculties such as creativity, self-improvement, and language use.

Enthusiasm

It was astonishing whenever a computer did anything remotely clever, John McCarthy referred to this period as the " **!**" era.

Lisp

John McCarthy defined in 1958 the high-level language , which was the dominant AI programming language for 30 years.

Machine evolution

, were based on the undoubtedly correct belief that by making an appropriate series of small mutations to a machine-code program, one can generate a program with good performance for any particular task.

1.4 Conclusion

Philosophers

made AI conceivable by considering the ideas that the mind is in some ways like a

Mathematicians

They set the groundwork for understanding computation and reasoning about .

Economists

formalized the problem of making decisions that maximize the expected to the decision maker.

Neuroscientists

discovered some facts about how the works and the ways in which it is similar to and different from computers

Psychologists

adopted the idea that humans and animals can be considered information processing

Computer engineers

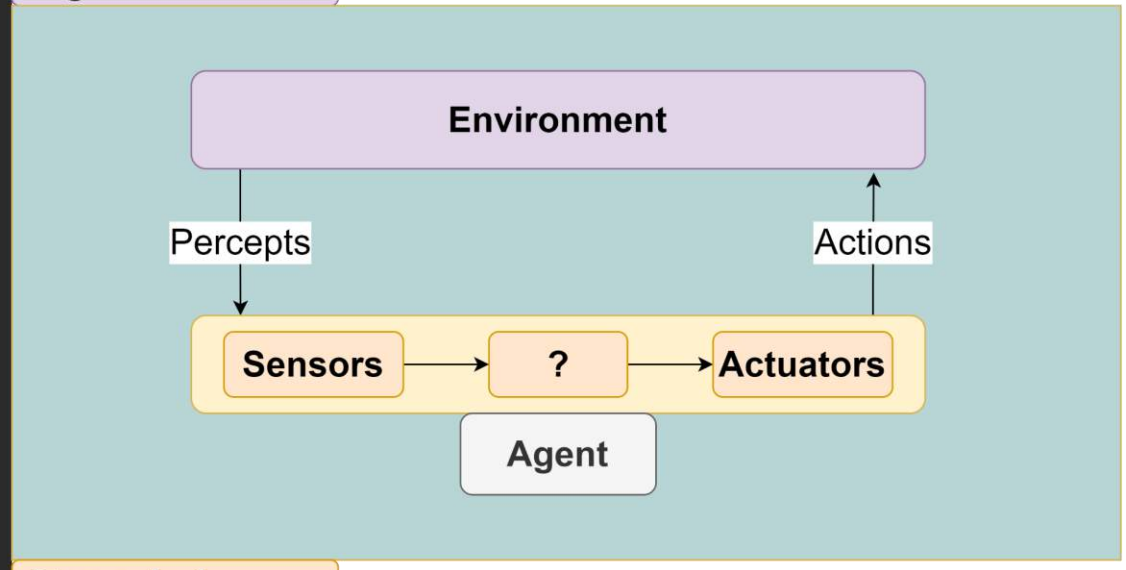
provided the ever-more-powerful that make AI applications possible.

2.1 Agents and Environments

1 Agent

Is anything that can **perceive** its **environment** trough **sensors** and acting upon that environment trough **actuators**.

Figure 1



Example 1

A human could be an agent, one which has for **sensors** ears and eyes, and for **actuators** legs and hands.

Example 2

A software agent receives key pushes, file contents and web packages as sensory inputs and acts on the environment displaying on the screen, writing files, etc.

2.1 Agents and Environments

Definitions

Percepts

Refer to the agent perceptual **inputs** at any given instant

Percepts sequence

Is the complete history of **everything** the agent has ever perceived

Agent function

Is the agent behavior explained in **mathematics**, the agent function **maps** any given percept sequence to an **action**

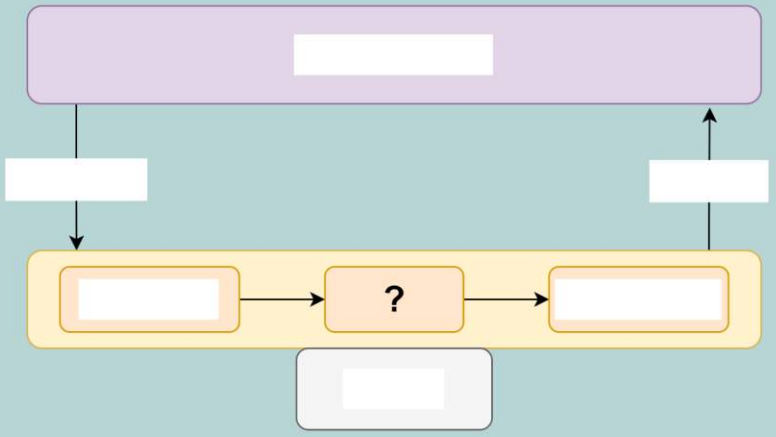
Is an abstract mathematical description; the agent program is a concrete implementation, running within the same physical system

2.1 Agents and Environments

1 Agent

Is anything that can **perceive** its **environment** trough **sensors** and acting upon that environment trough **actuators**.

Figure 1



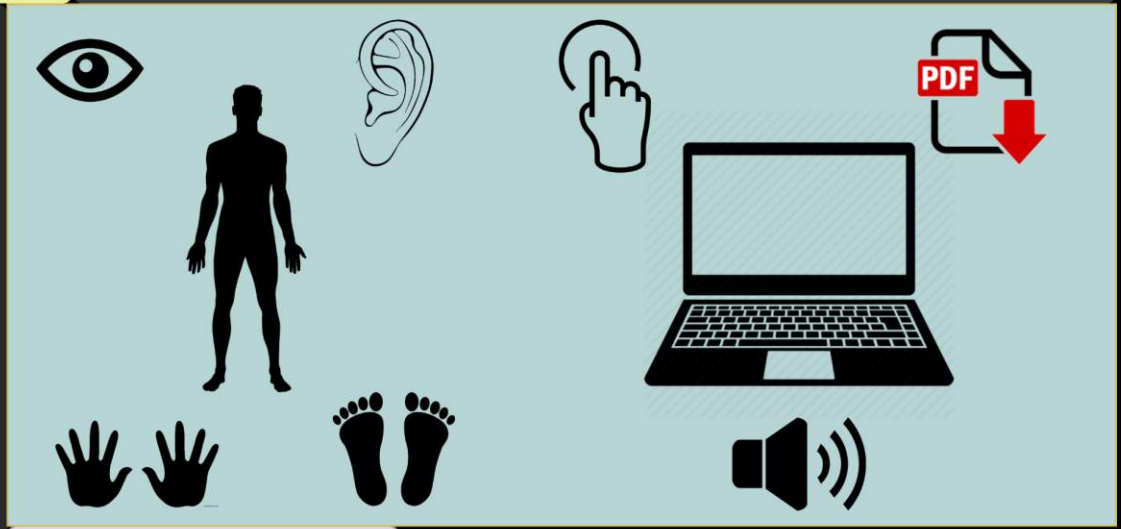
Example 1

A human could be an agent, one which has for _____ ears and eyes, and for _____ legs and hands.

Example 2

A software agent receives key pushes, file contents and web packages as sensory inputs and acts on the environment displaying on the screen, writing files, etc.

2.1 Agents and Environments



Definitions

Percepts

Refer to the agent perceptual _____ at any given instant

Percepts sequence

Is the complete history of _____ the agent has ever perceived

Agent function

Is the agent behavior explained in _____, the agent function _____ any given percept sequence to an _____

Is an abstract mathematical description; the agent program is a concrete implementation, running within the same physical system

2.2 Good behavior: the concept of rationality

1 Rational agent

is one that does the right thing, to test if an agent has good performance with the concept of **performance measurement**

2 Performance measurement

It is the evaluation of **environment states** to to see if its **desirable**

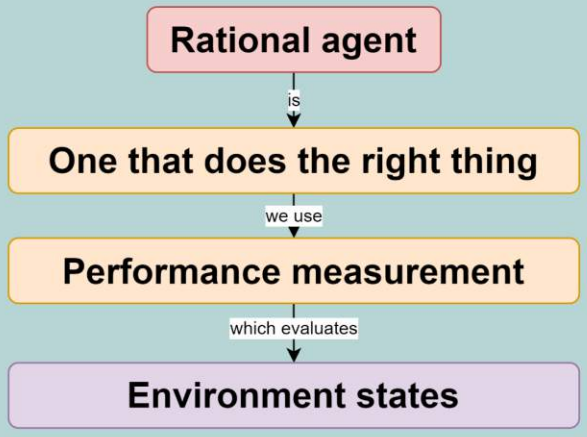
We do not define

Success in terms of agent opinion of its own performance, it is better to design performance measures according to what one actually wants in the **environment**

A philosophical question

Lets use for example an agent that cleans the floor, the average of cleanliness can be achieved by doing a mediocre job all the time or clean perfectly but take long breaks, ¿what is better?

Figure 1



2.3 Omniscience, learning, and autonomy

Imagine the scenario where we have two agents, both of them are in front a street and they need to cross, one of them already know if a car is gonna pass so when it cross nothing happens the other doesn't know, it just knows that it needs to look both ways to see if a car is coming, this concept is treated as **Omniscience** and **Rationality**.

3 Omniscience vs Rationality

We need to distinguish between these two, one knows the outcome of its action and acts accordingly, the other choose the looking action before stepping to the street.

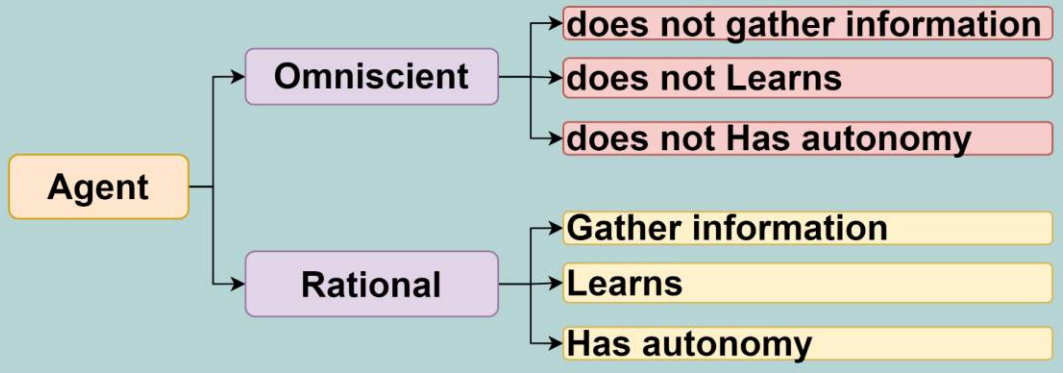
Information gathering

is an important part of rationality is doing actions in order to **modify** future precepts.

Learn

A rational agent needs to learn as much as possible from what it **perceives**, an agent that does not relies on its own precepts means that it lacks **autonomy**

Figure 1



2.2 Good behavior: the concept of rationality

1 Rational agent

is one that does the right thing, to test if an agent has good performance with the concept of _____ _____

2 Performance measurement

It is the evaluation of _____ _____ to to see if its _____.

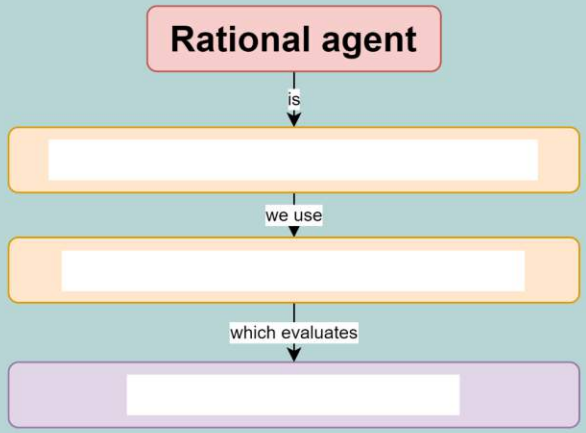
We do not define

Success in terms of agent opinion of its own performance, it is better to design performance measures according to what one actually wants in the _____.

A philosophical question

Lets use for example an agent that cleans the floor, the average of cleanliness can be achieved by doing a mediocre job all the time or clean perfectly but take long breaks, ¿what is better?

Figure 1



2.3 Omniscience, learning, and autonomy

Imagine the scenario where we have two agents, both of them are in front a street and they need to cross, one of them already know if a car is gonna pass so when it cross nothing happens the other doesn't know, it just knows that it needs to look both ways to see if a car is coming, this concept is treated as _____ and _____.

3 Omniscience vs Rationality

We need to distinguish between these two, one knows the outcome of its action and acts accordingly, the other choose the looking action before stepping to the street.

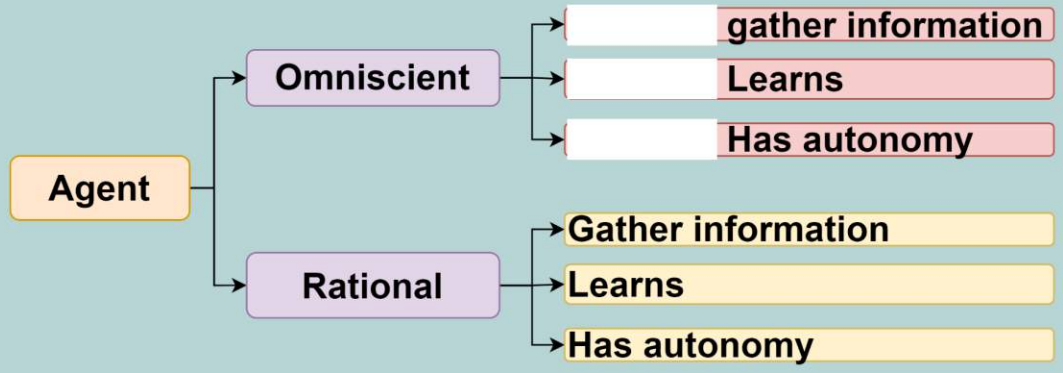
Information gathering

is an important part of rationality is doing actions in order to _____ future precepts.

Learn

A rational agent needs to learn as much as possible from what it _____, an agent that does not relies on its own precepts means that it lacks _____.

Figure 1



AgentSpeak

1 AgentSpeak

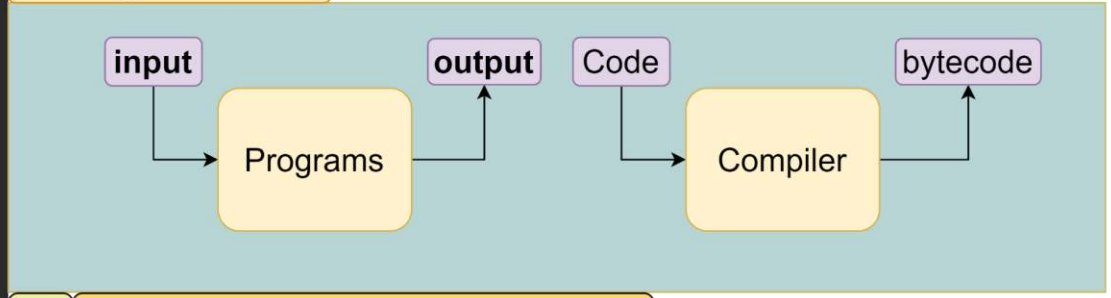
Is a **language** intended for developing **multi-agent** systems

1.1 Functional programs

this programs are threated as **functions**, from initial state to a terminal state.

Mathematically $f : I -> O$

Example



1.2 Reactive programs

these are programs that cannot be described by relational/functional view.

Example

Online banking systems, web servers and operating systems.

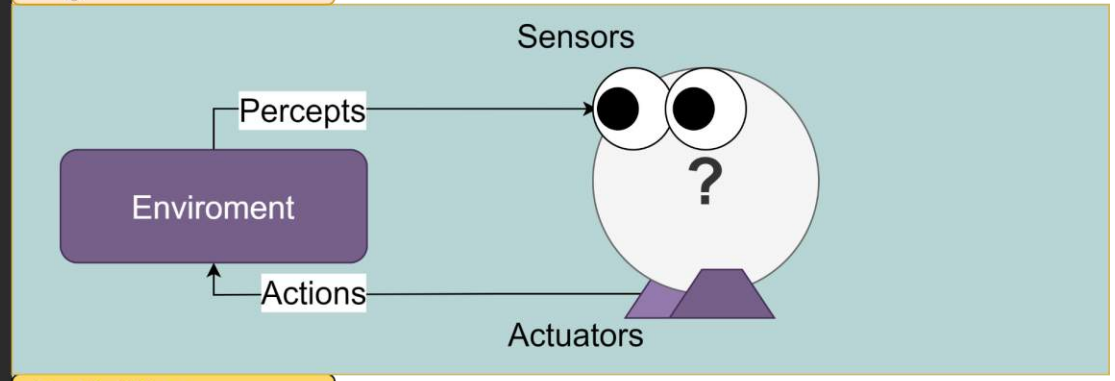
It is well know that these systems are much **harder** to make, even more complex is a subset of reactive systems called **agents**.

1.3 Agent

An Agent is a **reactive** system that has **autonomy**, this means we delegate a task and the system determines how best to achieve this task.

1 AgentSpeak

Figure 1.3



Definition

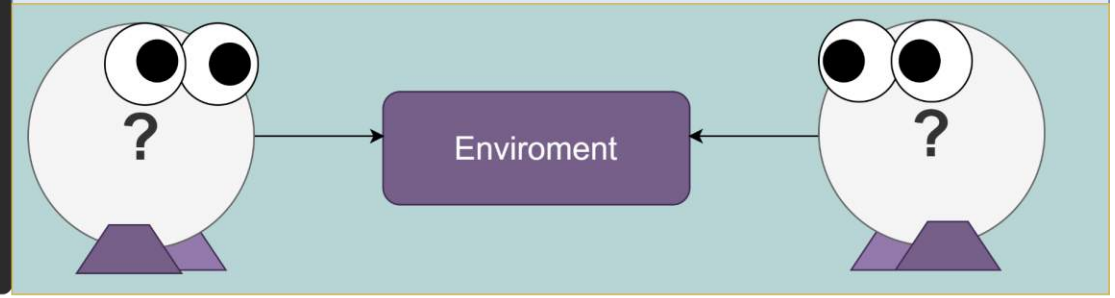
An **agent** is a system that are situated in some **environment**, this means that agents are capable of **sensing** their environment trough **sensors**, and have a list of possible **actions** that they can perform via **actuators** in order to modify their environment

To do

An agent achieve this by manipulating **plans**.

partial control

in almost all applications the agents have a **partial** control over their environment, because there will be other agents in the environment.



1 AgentSpeak

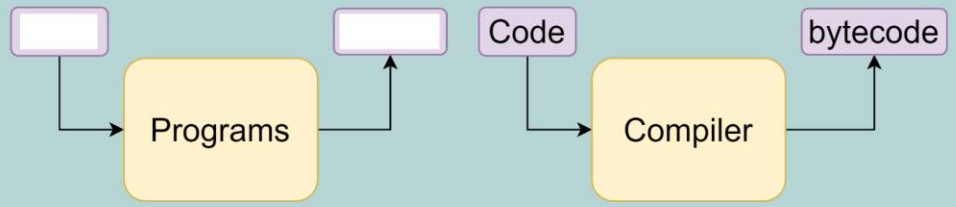
Is a _____ intended for developing _____ systems

1.1 Functional programs

this programs are threated as _____, from initial state to a terminal state.

Mathematically $f : I -> O$

Example



1.2 Reactive programs

these are programs that cannot be described by relational/functional view.

Example

Online banking systems, web servers and operating systems.

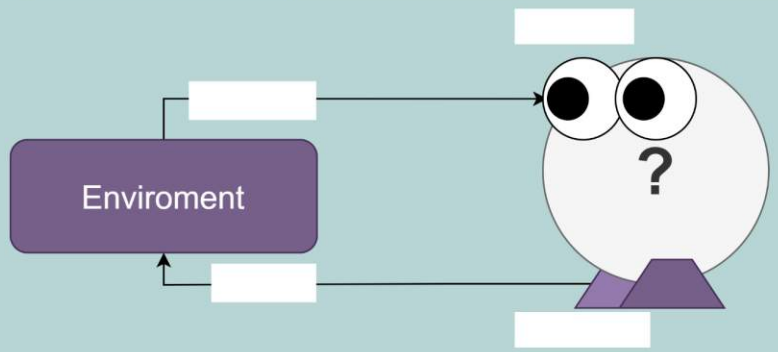
It is well know that these systems are much _____ to make, even more complex is a subset of reactive systems called _____.

1.3 Agent

An Agent is a _____ system that has _____, this means we delegate a task and the system determines how best to achieve this task.

1 AgentSpeak

Figure 1.3



Definition

An _____ is a system that are situated in some _____, this means that agents are capable of _____ their environment trough _____, and have a list of possible _____ that they can perform via _____ in order to modify their environment

To do

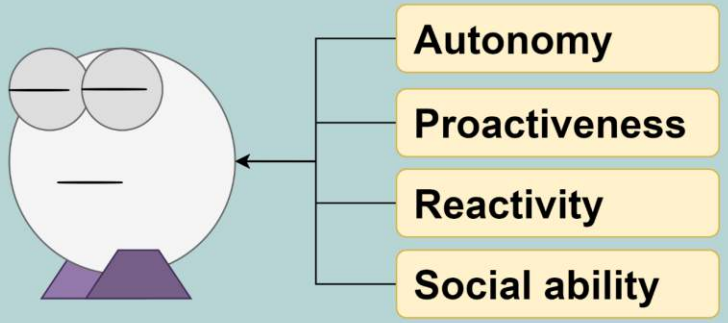
An agent achieve this by manipulating _____.

partial control

in almost all applications the agents have a _____ control over their environment, because there will be other agents in the environment.

2 Autonomy

An agent should have this 4 properties



2.1 Autonomy

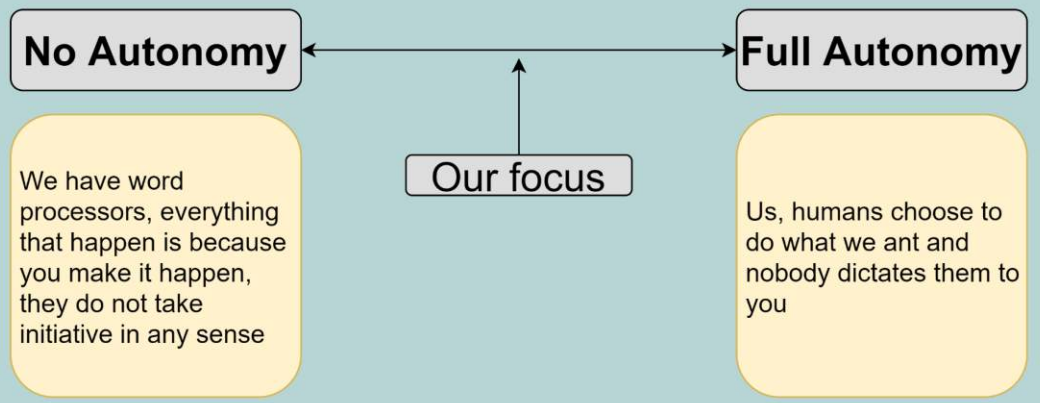
We want to delegate **goals** to agents which will decide how best to act.

Definition

Autonomy is being able to operate **independently** in order to achieve the **goals** we delegate to an agent.

Figure 1.3

we must recognize that autonomy is a very **broad** spectrum



2 AgentSpeak

2.2 Proactiveness

Proactiveness means being able to exhibit goal-directed **behavior**. If an agent has been delegated a particular goal, then we expect the agent to try to **achieve** this goal.

2.2 Reactiveness

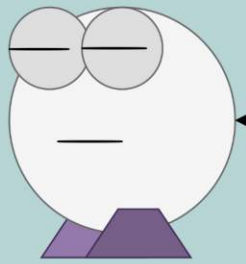
Means being **responsive** to change in the environment

2.3 Social ability

we talk about the ability of agents to **cooperate** and **coordinate** activities with other **agents**.
we want agents to be able to communicate their beliefs, goals and plans to one another.

2 Autonomy

An agent should have this 4 properties



2.1 Autonomy

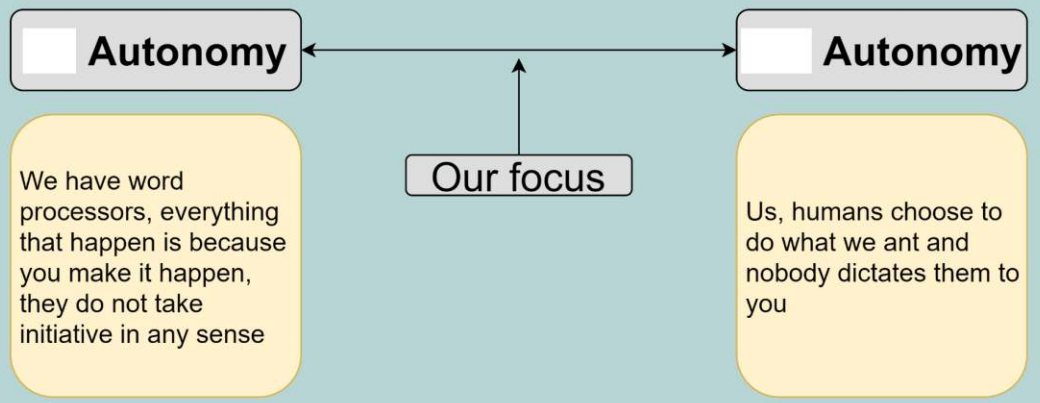
We want to delegate _____ to agents which will decide how best to act.

Definition

Autonomy is being able to operate _____ in order to achieve the **goals** we delegate to an agent.

Figure 1.3

we must recognize that autonomy is a very _____ spectrum



2 AgentSpeak

2.2 Proactiveness

Proactiveness means being able to exhibit goal-directed _____. If an agent has been delegated a particular goal, then we expect the agent to try to _____ this goal.

2.2 Reactiveness

Means being _____ to change in the environment

2.3 Social ability

we talk about the ability of agents to _____ and _____ activities with other _____.
we want agents to be able to communicate their beliefs, goals and plans to one another.

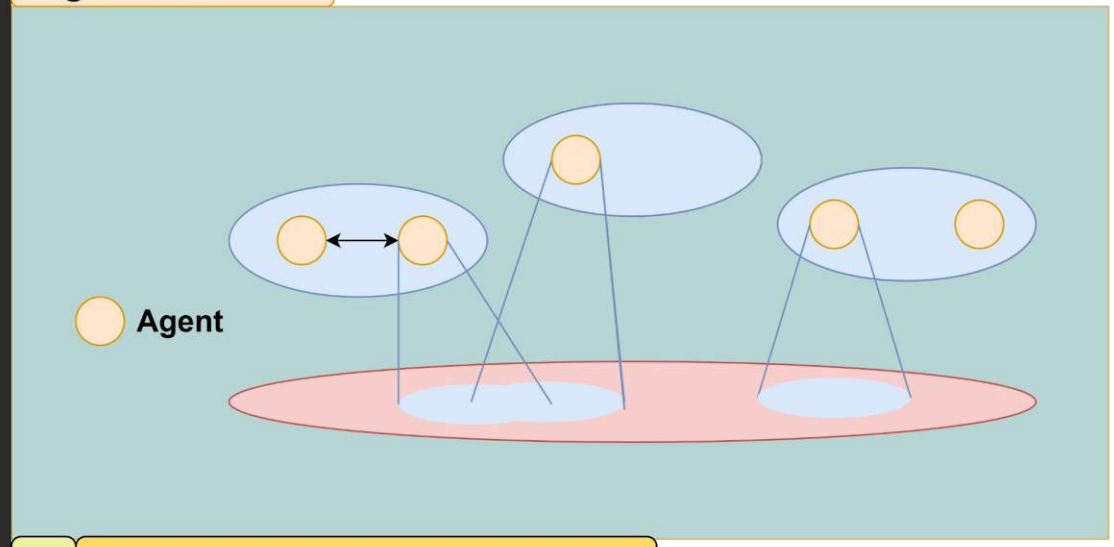
3 Multi-agent systems

In practice **single** agent systems are rare, the more common case is for agents to inhabit an environment which contains other **agents**.

3.1 Environment jointly controlled

Is an environment that has agents with spheres of influence **overlapping**

Figure 3.1



3.2 Hello world

What you will see next constitutes the definition of a single agent

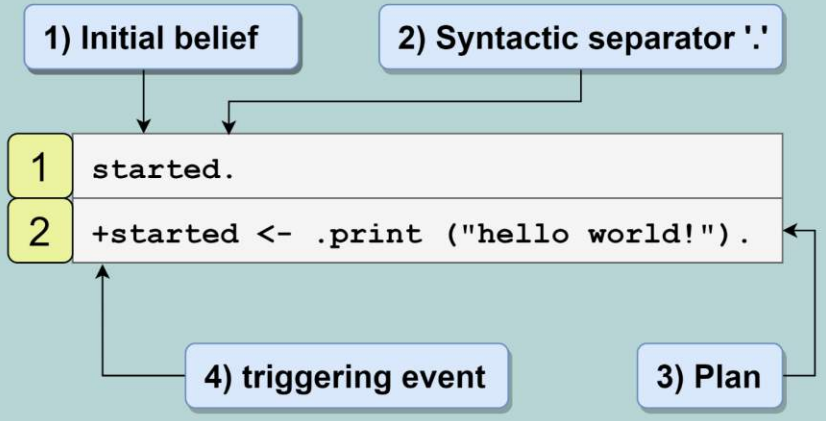
```
 started.
+started <- .print ("hello world!").
```

This will be saved in a file with extension **.asl**

3 Multi-agent systems

Definiton

The definition on the agent consists of two parts



1	this defines our initial belief four our agent, beliefs represent the **information** that the agent has currently been able to obtain about its **environment**.
2	'.' is a syntactic **separator**, much as a semi-color is in java or C.
3	This is the **plan** for the agent, which is in fact the only plan the agent has.
	it reads like this: whenever you come to believe **'started'**, print the text 'hello world!'.
4	the symbol '+' means , **'when you acquire the belief...'**

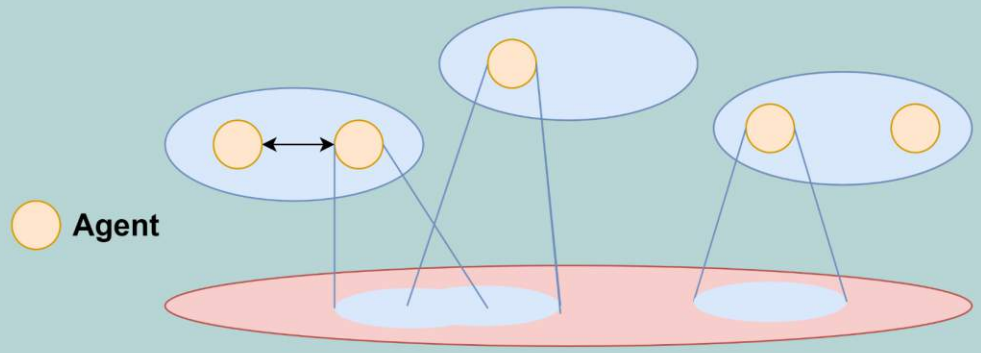
in sum, this plan will be triggered when the agent starts **executing**, because the trigger of a plan defines the events

In practice agent systems are rare, the more common case is for agents to inhabit an environment which contains other .

3.1 Environment jointly controlled

Is an environment that has agents with spheres of influence

Figure 3.1



3.2 Hello world

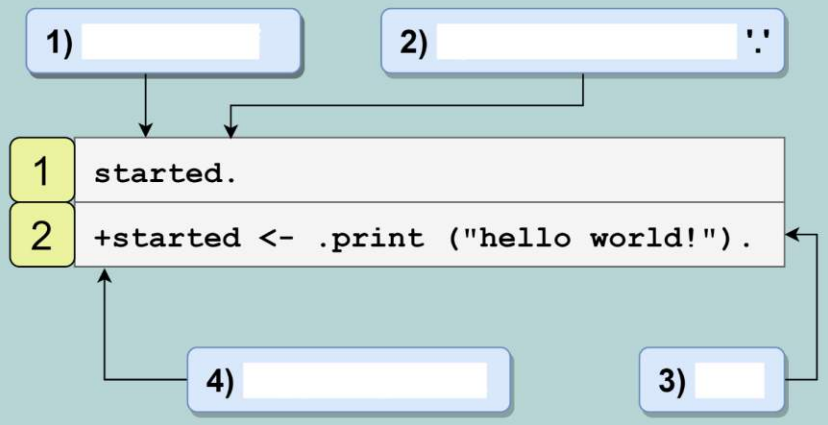
What you will see next constitutes the definition of a single agent

```
 started.
+started <- .print ("hello world!").
```

This will be saved in a file with extension .

Definiton

The definition on the agent consists of two parts



1	this defines our initial belief four our agent, beliefs represent the that the agent has currently been able to obtain about its .
2	'.' is a syntactic , much as a semi-color is in java or C.
3	This is the for the agent, which is in fact the only plan the agent has.
	it reads like this: whenever you come to believe ' ', print the text 'hello world!'.
4	the symbol '+' means , ' '

in sum, this plan will be triggered when the agent starts , because the trigger of a plan defines the events

4 Computing the factorial

the next code computes the factorial of 5, we will try to illustrate how loops work.

```
1  fact(0,1).
2  +fact(X,Y)
3      :X < 5
4      <- +fact(X+1, (X+1)* Y).
5  +fact(X,Y)
6      :X == 5
7      <- .print("fact 5 == ", Y).
```

1. the **initial** belief is added to its belief **set.**
2. whenever you acquire the **belief** that the factorial of X is Y
3. where X < 5
4. **add** to your beliefs the fact that the factorial of X + 1 is (X + 1) * Y
5. whenever you acquire the **belief** that the factorial of X is Y
6. where X == 5.
7. print out the value of Y.

4 Computing the factorial

this is a slightly more complex, but much efficient, version of the program.

```
1  !print_fact(5).
2  +!print_fact(N)
3    <- !fact(N,F);
4    .print("Factorial of ", N, " is ", F).
5  +!fact(N,1) : N == 0.
6  +!fact(N,F) : N  > 0
7    <- !fact(N-1, F1);
8    F = F1 * N.
```

1. Your **initial goal** is to print the **factorial** of 5.
2. whenever you acquire the **belief** that your goal is to print the factorial of **N**.
3. we first calculate the **factorial** of N.
4. and then printing out the **result**.
5. whenever you acquire the **goal** to calculate the factorial of N, which is 0, nothing else needs to be done.
6. whenever you acquire the **goal** to calculate the factorial of N, when N > 0.
7. we **first** calculate the factorial of N-1.
8. **then** multiply that value by **N** to get the factorial of **N.**

4 Computing the factorial

the next code computes the factorial of 5, we will try to illustrate how loops work.

```
1  fact(0,1).
2  +fact(X,Y)
3      :X < 5
4      <- +fact(X+1, (X+1)* Y).
5  +fact(X,Y)
6      :X == 5
7      <- .print("fact 5 == ", Y).
```

1. the belief is added to its belief .
2. whenever you acquire the that the factorial of X is Y
3. where X < 5
4. to your beliefs the fact that the factorial of X + 1 is (X + 1) * Y
5. whenever you acquire the that the factorial of X is Y
6. where X == 5.
7. print out the value of Y.

4 Computing the factorial

this is a slightly more complex, but much efficient, version of the program.

```
1  !print_fact(5).
2  +!print_fact(N)
3    <- !fact(N,F);
4    .print("Factorial of ", N, " is ", F).
5  +!fact(N,1) : N == 0.
6  +!fact(N,F) : N  > 0
7    <- !fact(N-1, F1);
8    F = F1 * N.
```

1. Your is to print the of 5.
2. whenever you acquire the that your goal is to print the factorial of .
3. we first calculate the of N.
4. and then printing out the .
5. whenever you acquire the to calculate the factorial of N, which is 0, nothing else needs to be done.
6. whenever you acquire the to calculate the factorial of N, when N > 0.
7. we calculate the factorial of N-1.
8. multiply that value by to get the factorial of .

5 The BDI Agent model

its a model that stands for **belief**, **intention**, **desires** (BDI)

Created by

the philosofer **Michael Bratman**

When, Where

in ***1980***s in Stanford researcher institute

Consists

On the **theory** of human practical reasoning

Which focuses on

The roles of **intentions** in practical reasoning

5.1 Agent oriented programming

Computer programs have "**mental states**".

Which means

they have computation **resembles** to beliefs/ desires/ intentions

Definitions

Belief

Is the **information** that an agent know about the world

Desires

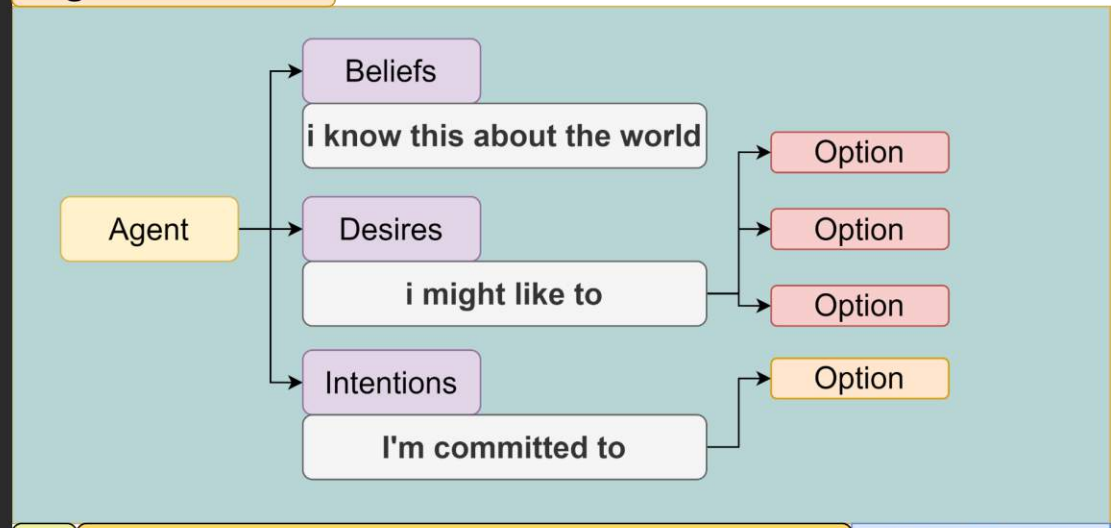
is the **options** that an agent might like to accomplish

Intentions

Is the options that are choose and is **committed** to do

5 The BDI Agent model

Figure 5.1



5.2 Agent-oriented programming (AOP)

is **programming** a computer with systems in terms of mentalistic notions such as belief, desire and intention.

Created by

Yoav Shoham in his work agent-oriented programming

Is

The **key** component of BDI model

5 The BDI Agent model

its a model that stands for _____ , _____ , _____ (BDI)

Created by

the philosofer _____

When, Where

in _____ s in Stanford researcher institute

Consists

On the _____ of human practical reasoning

Which focuses on

The roles of _____ in practical reasoning

5.1 Agent oriented programming

Computer programs have " _____ ".

Which means

they have computation _____ to beliefs/ desires/ intentions

Definitions

Belief

Is the _____ that an agent know about the world

Desires

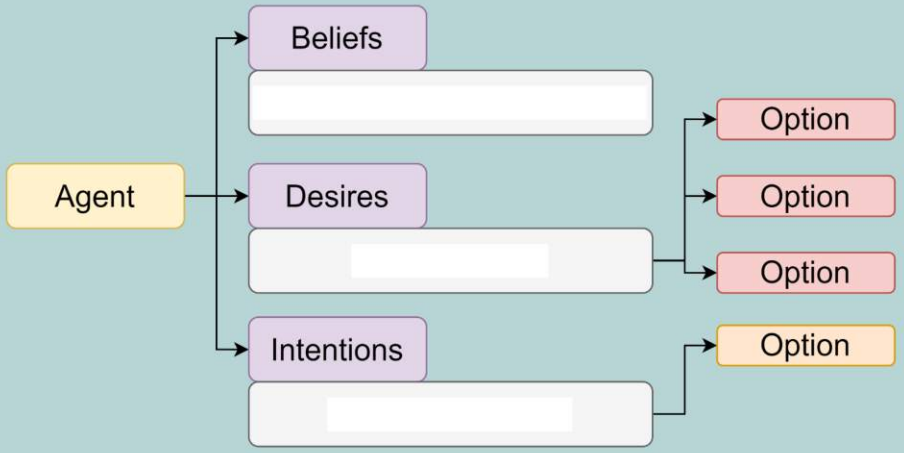
is the _____ that an agent might like to accomplish

Intentions

Is the options that are choose and is _____ to do

5 The BDI Agent model

Figure 5.1



5.2 Agent-oriented programming (AOP)

is _____ a computer with systems in terms of mentalistic notions such as belief, desire and intention.

Created by

_____ in his work agent-oriented programming

Is

The _____ component of BDI model

6 Practical reasoning

Figure 6.1

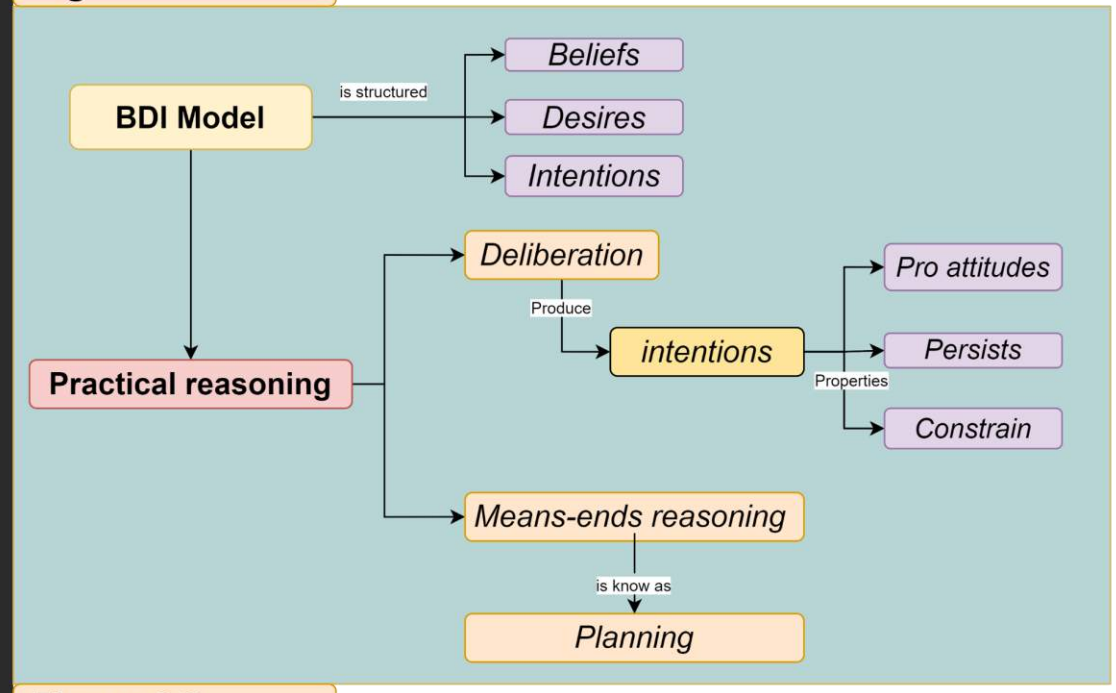
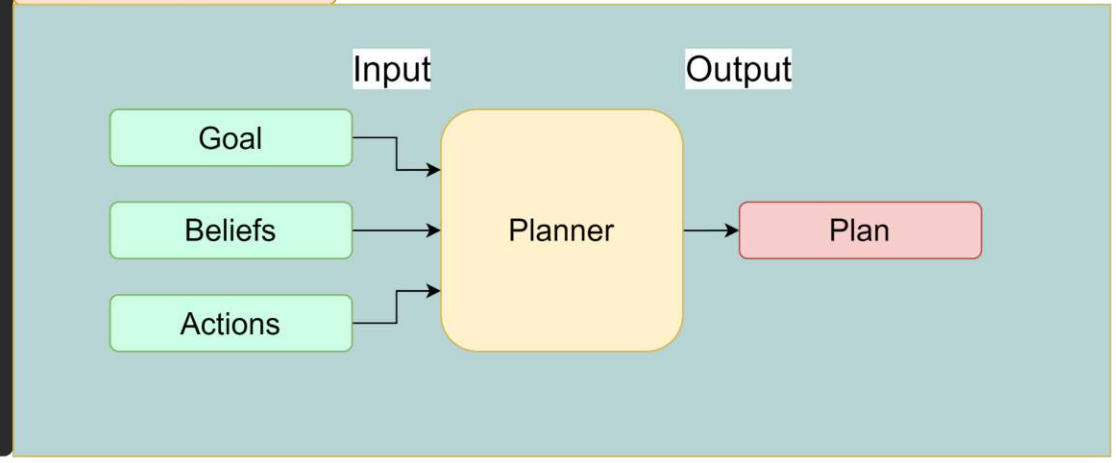


Figure 6.2



6 Practical reasoning

6.1 Practical reasoning

Is the ***decision-making*** of the BDI model, is reasoning directed towards actions.

Consists of

1 Deliberation

is a **process** that results in an agent adopting **intentions.**

Has properties

A Is proattitude

They tend to ***lead*** to action.

B Is persisting

you should persists with the intention and ***attempt*** to achieve it

C Is constraining

if you hold a particular intention you will not ***focus*** on other one

2 Means-ends reasoning

is the process of deciding ***how*** to achieve and end, using the available means

Is know as

Planning

Planner

is a system that takes ***input*** as intentions, goals, actions and generates a ***plan*** as output

Figure 6.1

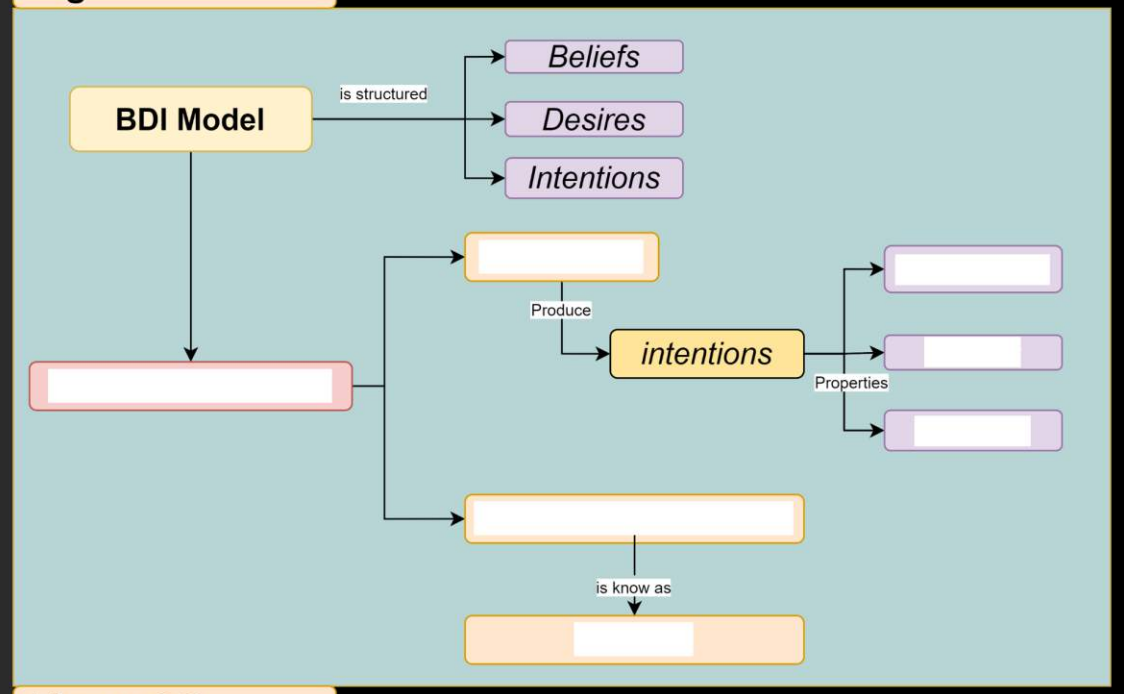
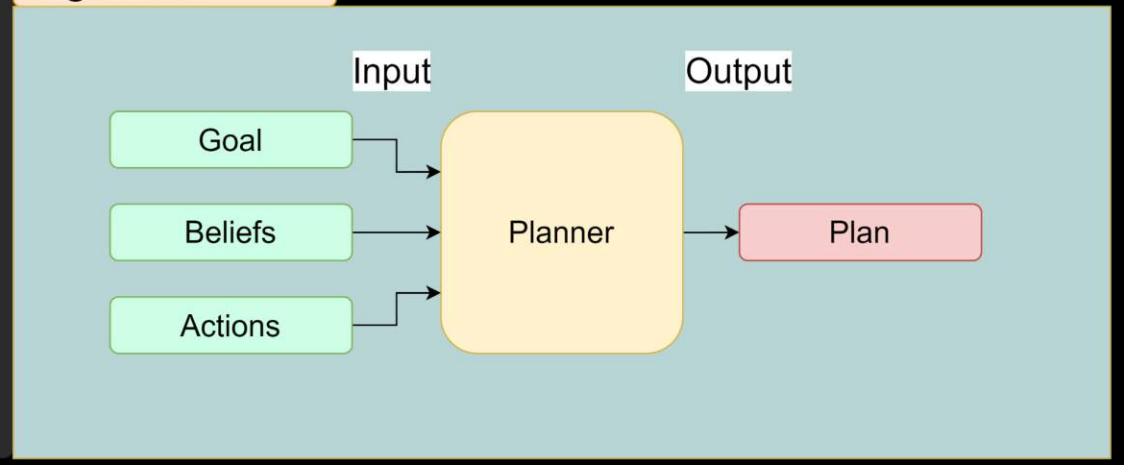


Figure 6.2



6.1 Practical reasoning

Is the _____ of the BDI model, is reasoning directed towards actions.

Consists of

1 Deliberation

is a _____ that results in an agent adopting _____.

Has properties

A Is proattitude

They tend to _____ to action.

B Is persisting

you should persists with the intention and _____ to achieve it

C Is constraining

if you hold a particular intention you will not _____ on other one

2 Means-ends reasoning

is the process of deciding _____ to achieve and end, using the available means

Is know as

_____

Planner

is a system that takes _____ as intentions, goals, actions and generates a _____ as output

4

Computing the factorial

1	$B \leftarrow B_0$;	***B*** represent **information** that the agent has about its environment
2	$I \leftarrow I_0$;	***I*** is the **intentions** the agent has committed to
3	**while(true)**	we start the life of our agent
4	get next percept ρ via sensors;	the agent **observes** the environment to get the next precept
5	$B \leftarrow brf(B, \rho)$;	he **updates** his beliefs, via the **belief revision function** denoted by brf().
6	$D \leftarrow options(B, I)$;	the agent determines its options based of his currents **beliefs** and **intention**.
7	$I \leftarrow filter(B, D, I)$;	the agent **chooses** between these options, selecting some to become **intentions**.
8	$\pi \leftarrow plan(B, I, A_c)$;	generating a **plan** to achieve its intentions, via plan()
9	**while not(empty(π) || succeeded(I,B) || impossible(I,B))**	we will repeat π until the plan is **empty**
10	$\alpha \leftarrow$ **first element of** π	we take the first **step** of our plan
11	$execute(\alpha)$	we **execute** it
12	$\pi \leftarrow$ **tail of** π	we **remove** our last element of our plan
13	**observe environment to get next percept(ρ)**	the agent pauses to **observe** its environment
14	$B \leftarrow brf(B, \rho)$;	the agent **updates** its beliefs
15	**if** *reconsider(I,B)*	the agent **reconsider** if its worth keeping the same intentions
16	$D \leftarrow options(B, I)$;	the agent **determines** its options based of his currents beliefs and intention.
17	$I \leftarrow filter(B, D, I)$;	the agent **chooses** between these options, selecting some to become intentions.
18	**if not** *sound(π,I,B)*	if the agent **believes** the plan is no longer a sound one
19	$\pi \leftarrow$ **plan** (B, I, A_c)	then it **re-plans**

4

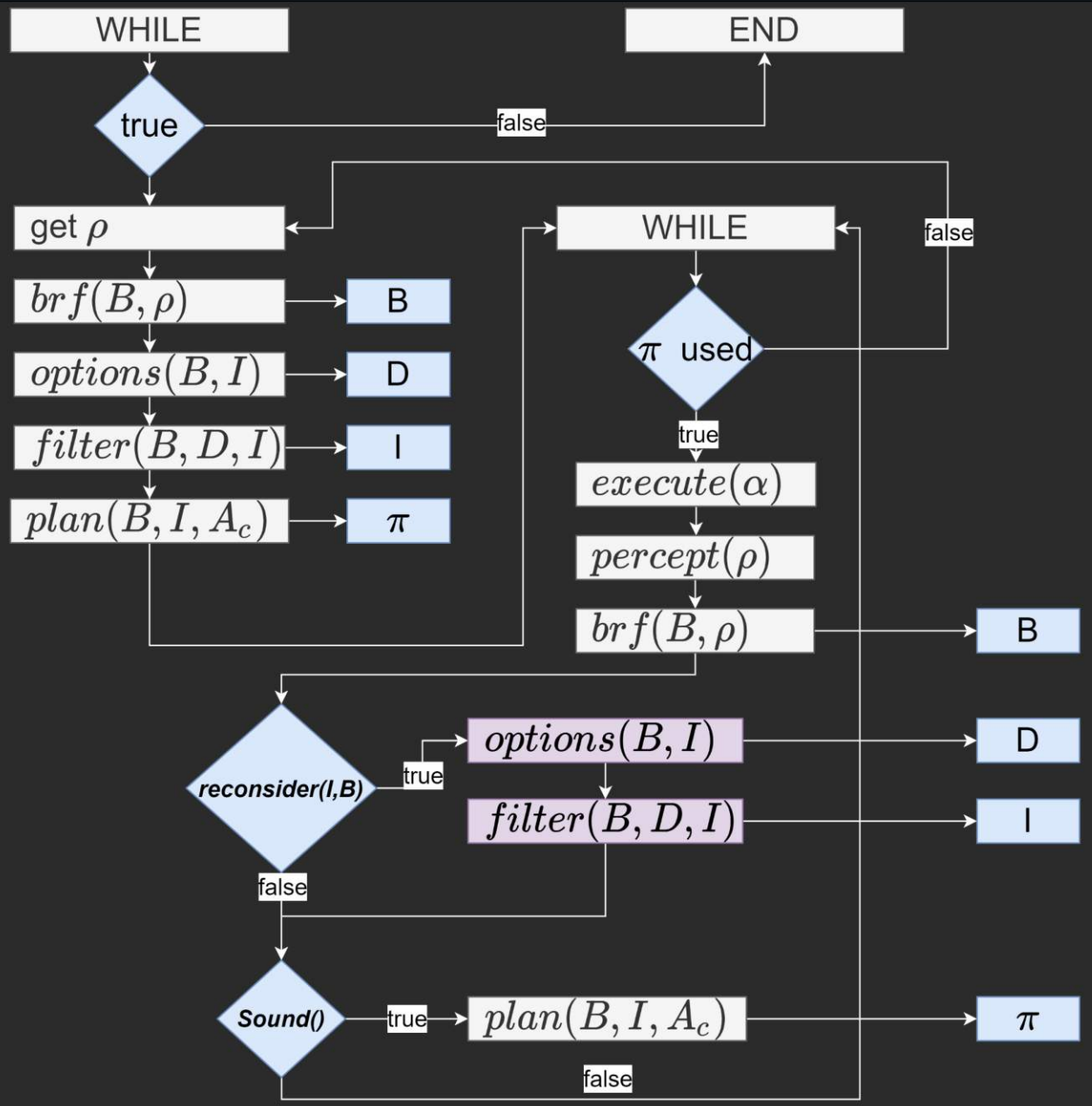
Computing the factorial

```
B ← B0;
I ← I0;
while(true)
    get next percept ρ via sensors;
    B ← brf(B, ρ);
    D ← options(B, I);
    I ← filter(B, D, I);
    π ← plan(B, I, Ac);
    while not(empty(π) || succeeded(I,B) || impossible(I,B))
        α ← first element of π
        execute(α)
        π ← tail of π
        observe environment to get next percept(ρ)
        B ← brf(B, ρ);
        if reconsider(I,B)
            D ← options(B, I);
            I ← filter(B, D, I);
        if not sound(π,I,B)
        π ← plan (B, I, Ac)
```

1. ***B*** represent _____ that the agent has about its environment
2. ***I*** is the _____ the agent has committed to
3. we start the life of our agent
4. the agent _____ the environment to get the next precept
5. he _____ his beliefs, via the _____ denoted by brf().
6. the agent determines its options based of his currents _____ and _____.
7. the agent _____ between these options, selecting some to become intentions.
8. generating a plan to achieve its intentions, via plan()
9. we will repeat π until the plan is _____
10. we take the first _____ of our plan
11. we _____ it
12. we _____ our last element of our plan
13. the agent pauses to _____ its environment
14. the agent _____ its beliefs
15. the agent _____ if its worth keeping the same intentions
16. the agent _____ its options based of his currents beliefs and intention.
17. the agent _____ between these options, selecting some to become intentions.
18. if the agent _____ the plan is no longer a sound one
19. then it _____

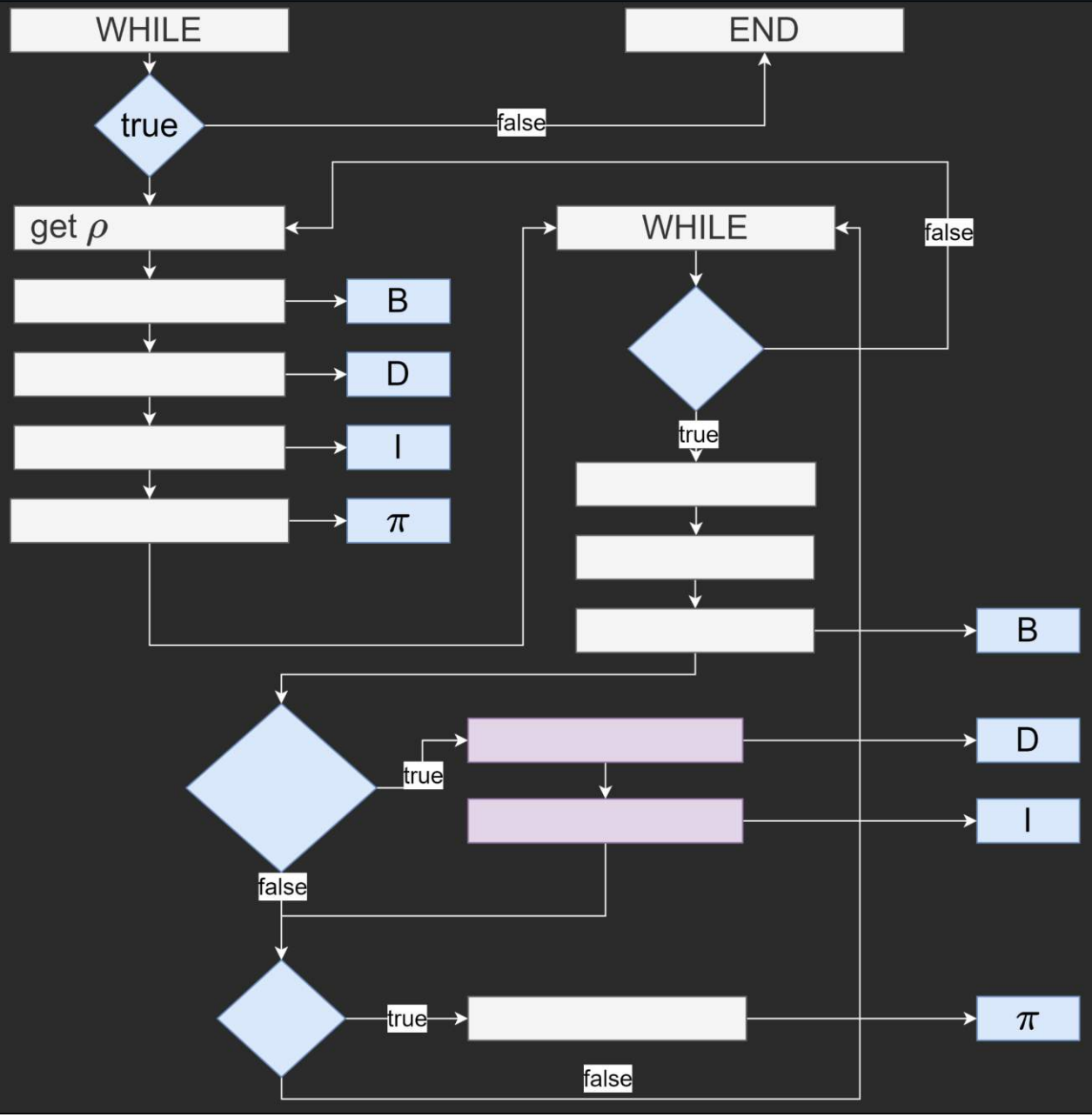
Computing the factorial

```
B ← B_0;
I ← I_0;
while(true)
    get next percept ρ via sensors;
    B ← brf(B, ρ);
    D ← options(B, I);
    I ← filter(B, D, I);
    π ← plan(B, I, A_c);
    while not(empty(π) || succeeded(I,B) || impossible(I,B))
        α ← first element of π
        execute(α)
        π ← tail of π
        observe environment to get next percept(ρ)
        B ← brf(B, ρ);
        if reconsider(I,B)
            D ← options(B, I);
            I ← filter(B, D, I);
        if not sound(π,I,B)
        π ← plan (B, I, A_c)
```



Computing the factorial

```
B ← B0;
I ← I0;
while(true)
    get next percept ρ via sensors;
    B ← brf(B, ρ);
    D ← options(B, I);
    I ← filter(B, D, I);
    π ← plan(B, I, Ac);
    while not(empty(π) || succeeded(I,B) || impossible(I,B))
        α ← first element of π
        execute(α)
        π ← tail of π
        observe environment to get next percept(ρ)
        B ← brf(B, ρ);
        if reconsider(I,B)
            D ← options(B, I);
            I ← filter(B, D, I);
        if not sound(π,I,B)
        π ← plan (B, I, Ac)
```



What is it

Is the first agent **architecture** to embody the BDI paradigm, it is also the most durable to develop agents.

Developed in

Standford research **institute**

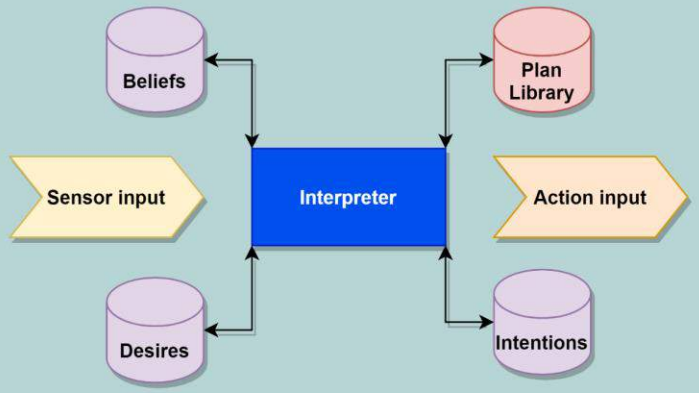
by

Michael georgeff and **Lansky**

Consists of

A library of **precompile** plans, which are manually constructed in advance, by the programmer.

Figure 9.1

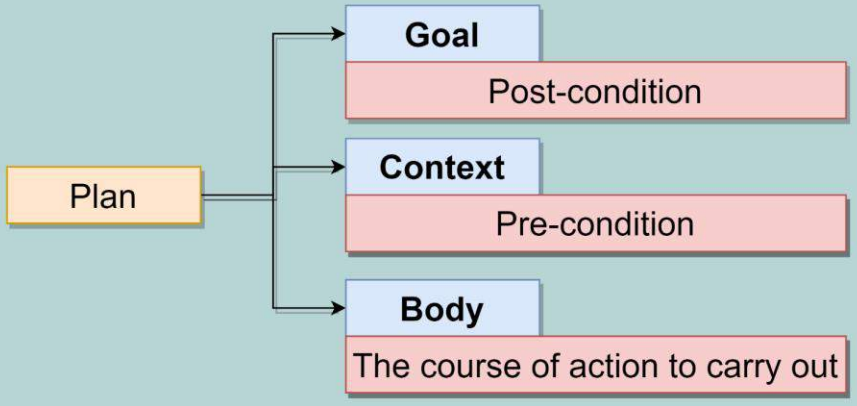


Plans in PRS

Plans in PRS each have the following components

9 Procedural reasoning system (PRS)

Figure 9.2



Goal

Is the things that it **achieves** by the plan, is the post-condition of the plan

Context

defines what must be true of the **environment** in order for the plan to be succesful.

Body

the body is slightly unusual, we assumed that a plan is a was simply a list of actions, but this can have more **goals** and **plans**.

AgentSpeak

Represents an **attempt** to distill the key features of the PRS into a simple, unified programming language.

What is it

Is the first agent _____ to embody the BDI paradigm, it is also the most durable to develop agents.

Developed in

Standford research _____

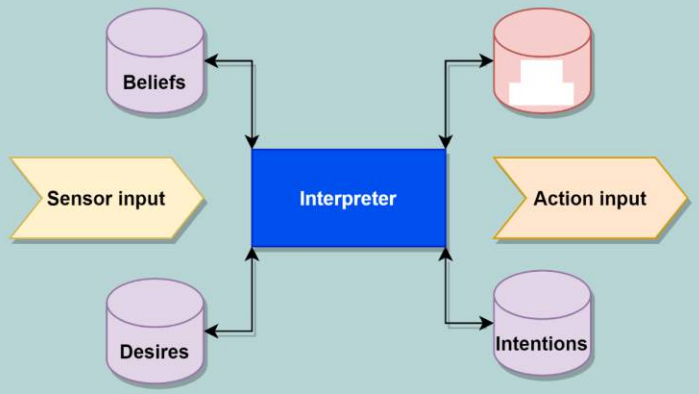
by

Michael georgeff and _____

Consists of

A library of _____ plans, which are manually constructed in advance, by the programmer.

Figure 9.1

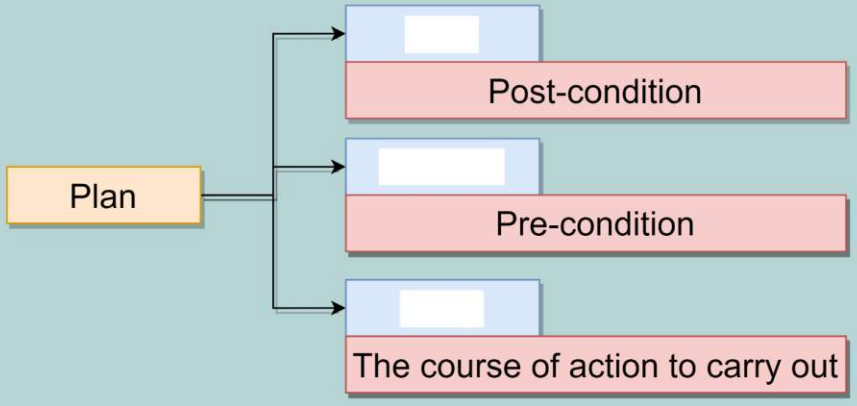


Plans in PRS

Plans in PRS each have the following components

9 Procedural reasoning system (PRS)

Figure 9.2



Goal

Is the things that it _____ by the plan, is the post-condition of the plan

Context

defines what must be true of the _____ in order for the plan to be successful.

Body

the body is slightly unusual, we assumed that a plan is a was simply a list of actions, but this can have more _____ and _____.

AgentSpeak

Represents an _____ to distill the key features of the PRS into a simple, unified programming language.

10 The jason agent programming language

Jason is an **extension** of **AgentSpeak**, which is based on the BDI arquitecture

We should make the distinction between

1 **Agent architecture**

software **framework** from where an agent program run for example the procedural reasoning system (**PRS**)

2 **Agent program**

are the **plans** that inhabits the architecture

1.1 **Reactive planning systems**

Systems that are not meant to compute the value of a function and terminate, but rather designed to be **permanently** running, reacting to some form of events.

They react

by executing **plans**

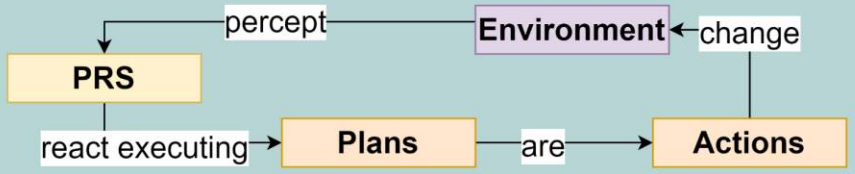
Plans are

Courses of **actions** that agents commit to do, as a way to handle such events

Actions are

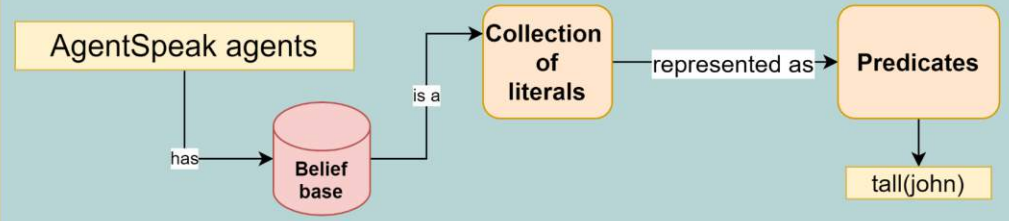
changes to the **environment**, in such a way that we expect the agents goals to be achieved.

Figure 10.1



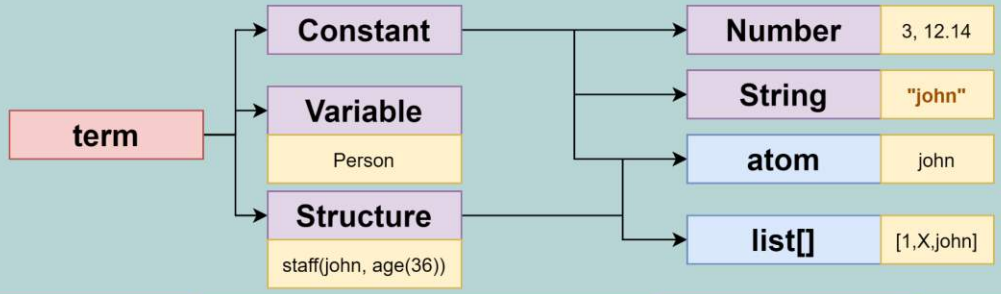
10 The jason agent programming language

Figure 10.2



john being tall	**tall(john)**
john likes music	**likes(john, music)**

Figure 10.3



atom

Is any symbol starting with a **lower** case letter, is used to represent particular individuals or objects, it is also classified as a constant.

Variable

Is a symbol starting with an **uppercase** letter, initially they are uninstantiated and once instantiated to a particular value, they maintain the value through their scope, that is the plan.
A variable could be bound to a constant or structure.

term

is used to **refer** to a constant, variable or a structure.

structure

Is used to represent complex data, structures **start** with an atom.

list

Is a type of structure that starts with **[]**, **|** can be used to simplify, for example [1,2,3] with [1|3].

Jason is an _____ of **AgentSpeak**, which is based on the BDI architecture

We should make the distinction between

1 **Agent architecture**

software _____ from where an agent program run for example the procedural reasoning system (____)

2 **Agent program**

are the _____ that inhabits the architecture

1.1 **Reactive planning systems**

Systems that are not meant to compute the value of a function and terminate, but rather designed to be _____ running, reacting to some form of events.

They react

by executing _____

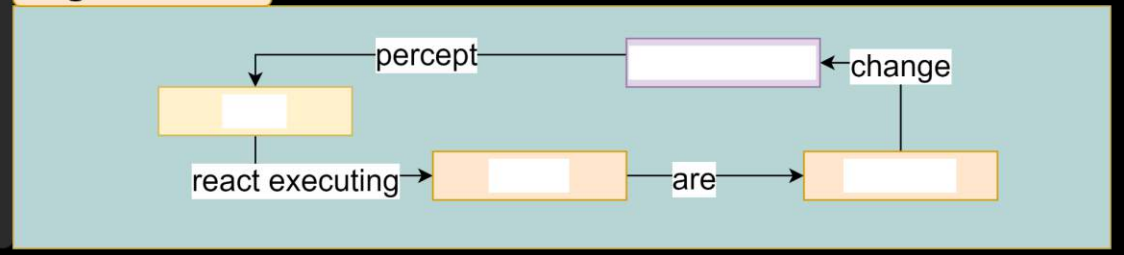
Plans are

Courses of _____ that agents commit to do, as a way to handle such events

Actions are

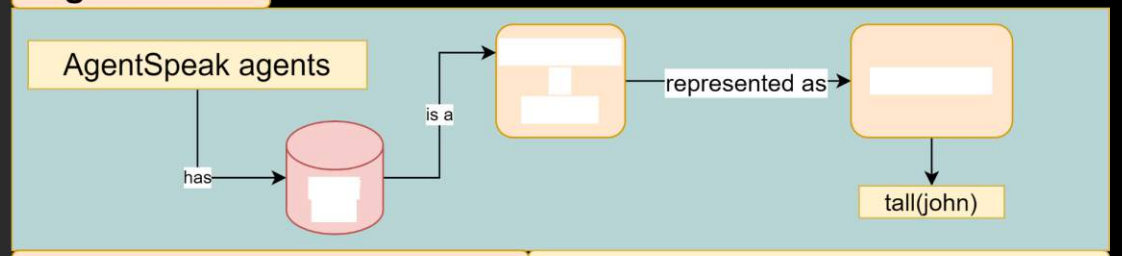
changes to the _____, in such a way that we expect the agents goals to be achieved.

Figure 10.1



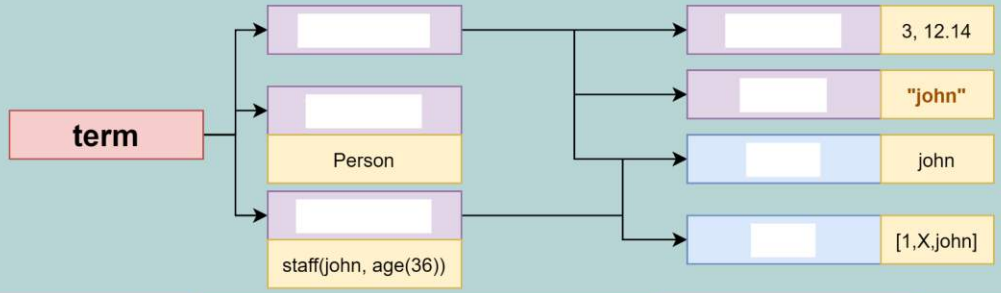
10 The jason agent programming language

Figure 10.2



john being tall	**tall(john)**
john likes music	**likes(john, music)**

Figure 10.3



atom

Is any symbol starting with a _____ case letter, is used to represent particular individuals or objects, it is also classified as a constant.

Variable

Is a symbol starting with an _____ letter, initially they are uninstantiated and once instantiated to a particular value, they maintain the value through their scope, that is the plan.
A variable could be bound to a constant or structure.

term

is used to _____ to a constant, variable or a structure.

structure

Is used to represent complex data, structures _____ with an atom.

list

Is a type of structure that starts with _____, _____ can be used to simplify, for example [1,2,3] with [1|3].

These are complex terms, that are enclosed in **square** brackets following a literal

Example

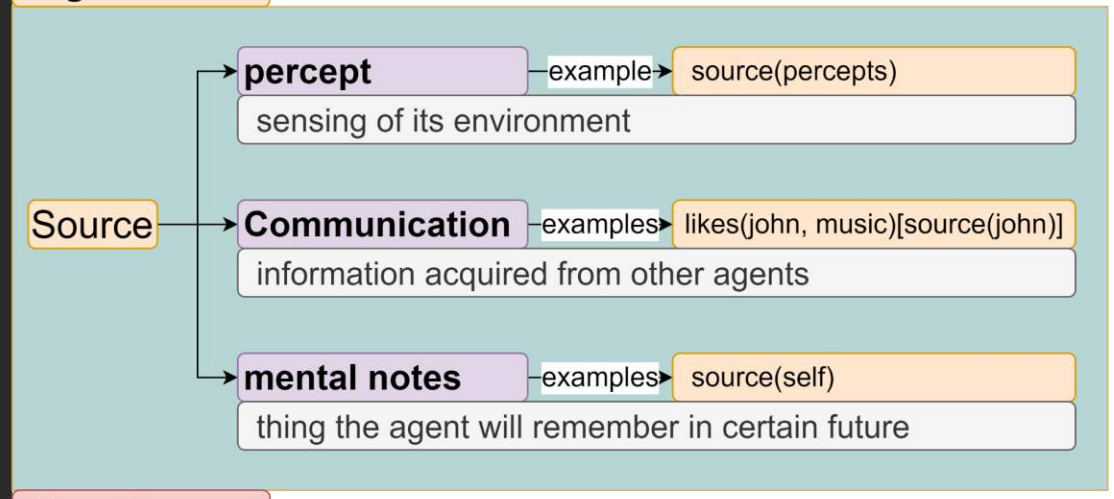
busy(john)[expires(autunm)]

expires does not mean **anything** to the jason interpreter

however

there are belief annotations, that have specific **meanings** for the interpreter

Figure 11.1



Negations

Is used to express that an agent explicitly believes something is **false**.

Example

color(box1, white)	the agent believes the color of box1 is white
~color(box1, white)	the agent believes the color of box1 is not white

Exercise **Maria beliefset**

color(box1, blue) [source(bob)]	agent bob informed maria that box 1 is blue
color(box1, white) [source(john)]	agent john informed maria that box1 is not white
color(box1, red) [source(percept)]	Maria perceive with a camera that box 1 is red
colorblind(bob) [source(self), degofCert(0.7)]	Maria notes to herself with a 0.7 that bob is color blind
lier(bob) [source(self), degofCert(0.2)]	Maria notes to herself with a 0.2 that bob is a liar

11.2 Rules

Rules allow us to conclude new things based on things we already know

Exercise

```
likely_color(C,B)
    :- color(C,B)[source(S)] & (S == self | S == percept)
```

The most likely color is either what which the agent notes to himself earlier, or the one that he perceives in the environment.

Some thoughts

All beliefs that appear in an agents source code are assumed to be '**mental notes**'.

Example

likes(john,music)	**likes(john,music) [source(self)]**

11 Annotations

These are complex terms, that are enclosed in brackets following a literal

Example

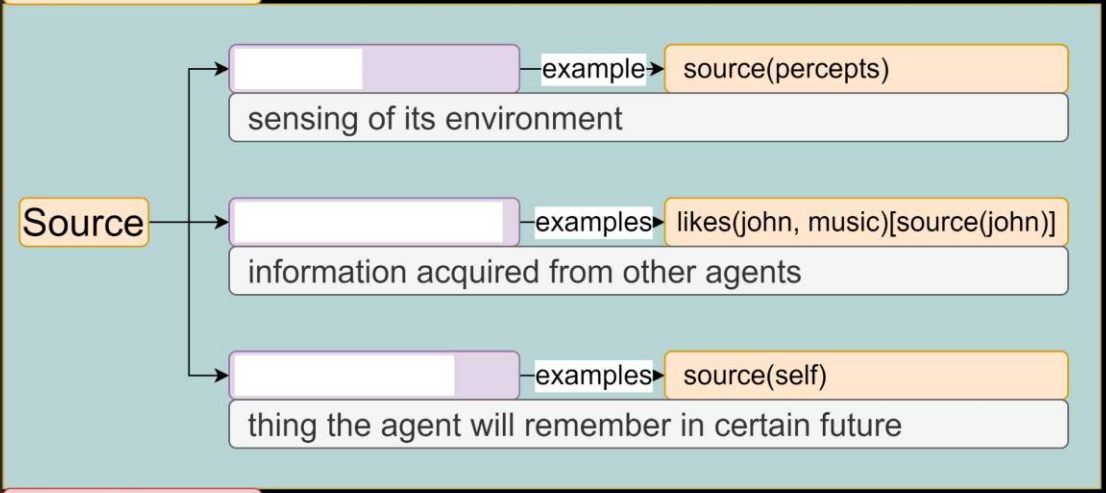
busy(john)[expires(autunm)]

expires does not mean to the jason interpreter

however

there are belief annotations, that have specific for the interpreter

Figure 11.1



Negations

Is used to express that an agent explicitly believes something is .

Example

	the agent believes the color of box1 is white
	the agent believes the color of box1 is not white

11 Annotations

Exercise **Maria beliefset**

	agent bob informed maria that box 1 is blue
	agent john informed maria that box1 is not white
	Maria perceive with a camera that box 1 is red
	Maria notes to herself with a 0.7 that bob is color blind
	Maria notes to herself with a 0.2 that bob is a liar

11.2 Rules

Rules allow us to conclude new things based on things we already know

Exercise

```
likely_color(C,B)
    :-
```

The most likely color is either what which the agent notes to himself earlier, or the one that he perceives in the environment.

Some thoughts

All beliefs that appear in an agents source code are assumed to be ' '.

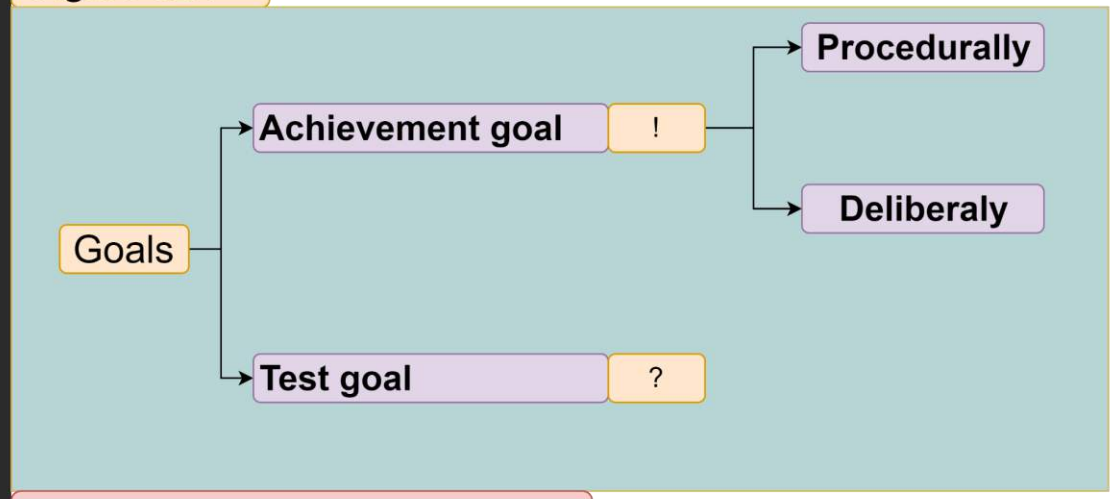
Example

likes(john,music)	

Goals

A **fundamental** notion, they express the **properties** of the states of the world the agent wishes to bring about.

Figure 12.1



Achievement Goal

Consider the example

!owns(house)

The agent will try to **achieve** a certain state of affairs until the agent indeed owns a house

Test Goal

Consider the example

?bank_balance(BB)

This will retrieve information that is available in the agent belief base, BB will be instantiated with the **money** the agent currently **believe** its bank base is.

Goals

Initial Goals

This are the goals that we provide for the agent in the **source code**, this is when the agent first starts running

Consider the example

!clean(house)

This agent would start running with the long term goal of getting the house **clean.**

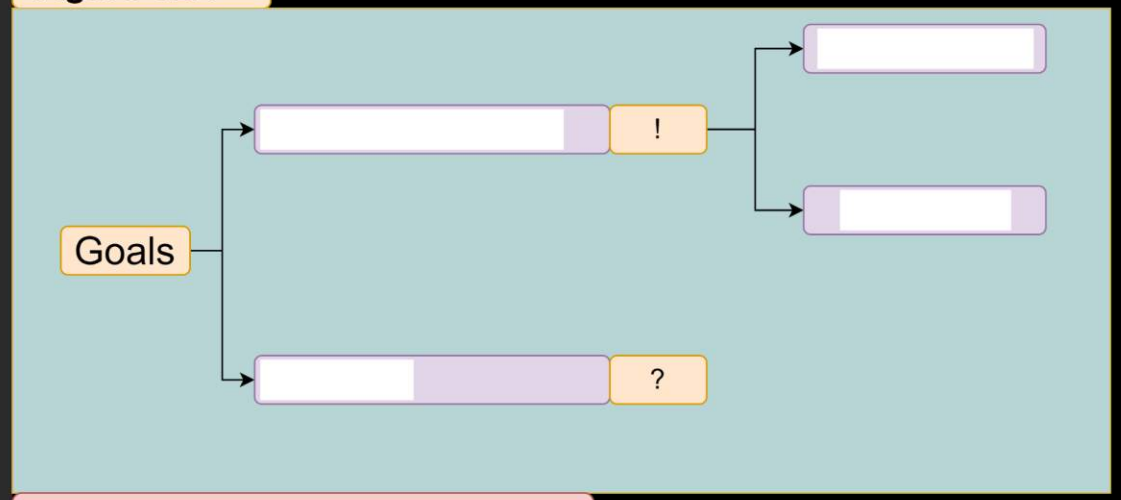
¿what is next?

the changes in the agent beliefs and goals **triggers** the execution of plans.

Goals

A _____ notion, they express the _____ of the states of the world the agent wishes to bring about.

Figure 12.1



Achievement Goal

Consider the example

!owns(house)

The agent will try to _____ a certain state of affairs until the agent indeed owns a house

Test Goal

Consider the example

?bank_balance(BB)

This will retrieve information that is available in the agent belief base, BB will be instantiated with the _____ the agent currently _____ its bank base is.

Goals

Initial Goals

This are the goals that we provide for the agent in the _____, this is when the agent first starts running

Consider the example

!clean(house)

This agent would start running with the long term goal of getting the house _____.

¿what is next?

the changes in the agent beliefs and goals _____ the execution of plans.

13 Plans

A plan is a course of action executed because of a **change** in the agent beliefs and goals.

Consists

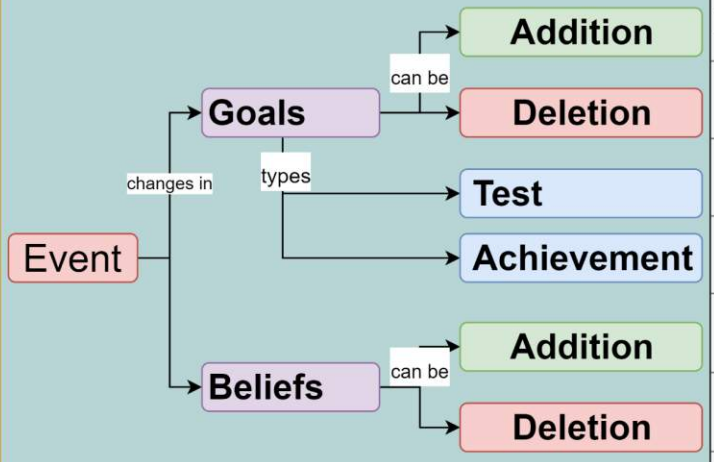
A plan consists of three parts separated by "**;**" and "**<-**".

Figure 13.1

1 Triggering event

Is a **change** that the agent is attentive for the environment

Figure 13.2



Notation	Name
+!	Belief addition
-!	Belief deletion
+!!	Goal addition
-!!	goal deletion
+?!	Test-goal additior
-?!	Test-goal deletior

13 Plans

2 Context

Is used to **evaluate** the current station and determine what plan is used, it can be compared to an ***if*** in programming.

Applicable context

Is a context **true** at the moment

Table 13.1

Syntax	Meaning
!	*the agent believes !! is true*
-!	*the agent believes !! is false*
not !	*the agent does not believe !! is false*
not -!	*the agent does not believe !! is false*

Examples

+!prepare(Something)
: number_of_people(N) & stock(Something, S) & S > N

when the agents comes to have the goal of preparing something, provided it has enough of it in stock for the number of people who will be eating.

+!buy(Something)
: not -legal(Something) & price(Something, P)
& bank_balance(B) & B > P

when the agent comes to have a new goal to buy something, provided it does not have explicit information saying is illegal to buy it, and it has enough money for the purchase.

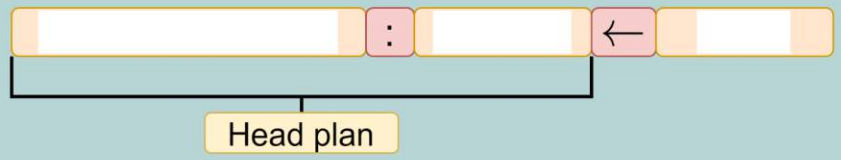
13 Plans

A plan is a course of action executed because of a in the agent beliefs and goals.

Consists

A plan consists of three parts separated by " " and " ".

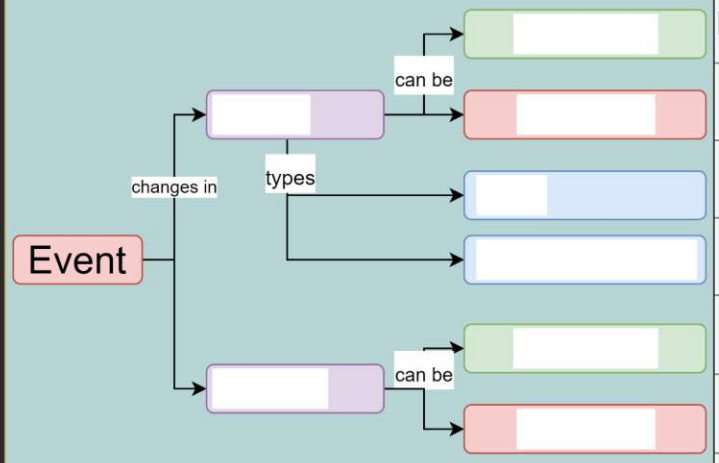
Figure 13.1



1 Triggering event

Is a that the agent is attentive for the environment

Figure 13.2



Notation	Name
	Belief addition
	Belief deletion
	Goal addition
	goal deletion
	Test-goal addition
	Test-goal deletion

13 Plans

2 Context

Is used to the current station and determine what plan is used, it can be compared to an in programming.

Applicable context

Is a context at the moment

Table 13.1

Syntax	Meaning
	the agent believes l is true
	the agent believes l is false
	the agent does not believe l is false
	the agent does not believe l is false

Examples

when the agents comes to have the goal of preparing something, provided it has enough of it in stock for the number of people who will be eating.

when the agent comes to have a new goal to buy something, provided it does not have explicit information saying is illegal to buy it, and it has enough money for the purchase.

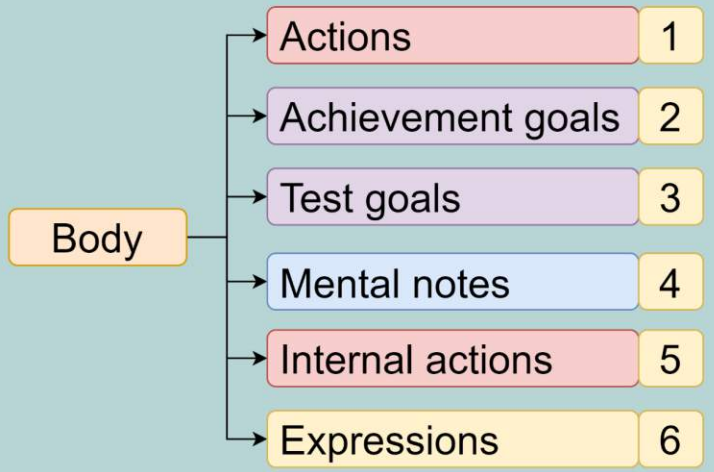
14 Body

Defines a course of action for the agent to take when an **event** has happened and the context of the plan is **true**.

is represented

by a sequence of formulae, each separated from the other by **";"**.

Figure 14.1



1 Actions

this is a simply **predicate** which is in the body, a simple predicate will be interpreted as an **action**.

Example

Rotate(Arm)

Tip

Is worth taking in to account that an action may not be able to complete, we can add a **boolean** of completion to notify if an action fails.

14 Body

2 Achievement goals

the representation is a **literal**, denoted with a prefix "**!**".

Consider

a1;
!g2;
a3;

The agent will do a1, then he will try to achieve g2, that is, executing other plan, and only when that plan finished, a3 will be executed.

Tip

we can skip this process by using '!!' operator.

Consider

a1;
!!g2;
a3;

The agent will do a3 as soon as the agents have a new goal.

3 Test goals

This are used to **retrieve** information from the belief base

Example

?Coords(Target, X, Y)

which will instantiate those variables with the information.

However

If for some reason the agent does not have the information required for the plan to be executed, this will make the plan **fail**.

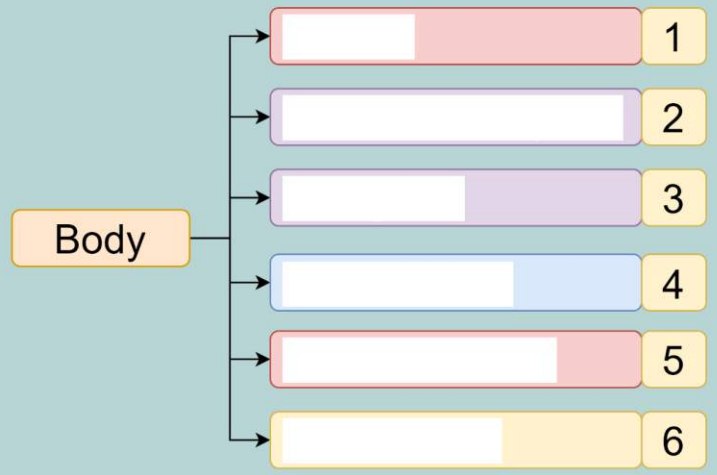
14 Body

Defines a course of action for the agent to take when an _____ has happened and the context of the plan is _____.

is represented

by a sequence of formulae, each separated from the other by " ".

Figure 14.1



1 Actions

this is a simply _____ which is in the body, a simple predicate will be interpreted as an _____.

Example

Rotate(Arm)

Tip

Is worth taking in to account that an action may not be able to complete, we can add a _____ of completion to notify if an action fails.

14 Body

2 Achievement goals

the representation is a _____, denoted with a prefix " ".

Consider

The agent will do a1, then he will try to achieve g2, that is, executing other plan, and only when that plan finished, a3 will be executed.

Tip

we can skip this process by using ' ' operator.

Consider

The agent will do a3 as soon as the agents have a new goal.

3 Test goals

This are used to _____ information from the belief base

Example

?Coords(Target, X, Y)

which will instantiate those variables with the information.

However

If for some reason the agent does not have the information required for the plan to be executed, this will make the plan _____.

4 Mental notes

are beliefs generated during the executing of a plan, they have a **source(self)** annotation to distinguish them within the belief base.

Example 4.1

```
+feed(dog, Today)
  feed(dog, Today)[Source(self)]
```

However

we normally have to **delete** mental notes that are no longer **necessary** such as.

```
~feed(dog, Today)
```

note that

attempting to delete a belief that does not exists is simply **ignored**, it does not cause a plan failure.

5 Internal actions

internal actions are actions executed **within** the agent, where as actions are outside the agent mind.

denoted

internal actions is distinguish from environment actions by the operator '.'

Example 5.1

```
.send(askif, raining)
.print("hello world")
```


6 Expressions

This are conditions that you can place in your **body's** plan

Example 6.1

```
X >= Y * 2
```

note that, if the condition is not meet the plan will **fail**

== equal	\== not equal
= unifier	=.. deconstruct a literal to a list.

7 Plan label

we can give a plan a specific **label**, we can address this label as the **name** of the plan.

by default

all plans have labels automatically **generated** by jason.

Figure 7.1

```
@label TrigerintEvent : context <- body
```

4 Mental notes

are beliefs generated during the executing of a plan, they have a _____ annotation to distinguish them within the belief base.

Example 4.1

+feed(dog, Today)

_____

However

we normally have to _____ mental notes that are no longer _____ such as.

_____

note that

attempting to delete a belief that does not exists is simply _____, it does not cause a plan failure.

5 Internal actions

internal actions are actions executed _____ the agent, where as actions are outside the agent mind.

denoted

internal actions is distinguish from environment actions by the operator ' '

Example 5.1

.send(askif, raining)
.print("hello world")

15 Body

6 Expressions

This are conditions that you can place in your _____ plan

Example 6.1

X >= Y * 2

note that, if the condition is not meet the plan will _____

== _____	**\==** _____
= _____	_____ **deconstruct a literal to a list.**

7 Plan label

we can give a plan a specific _____, we can address this label as the _____ of the plan.

by default

all plans have labels automatically _____ by jason.

Figure 7.1

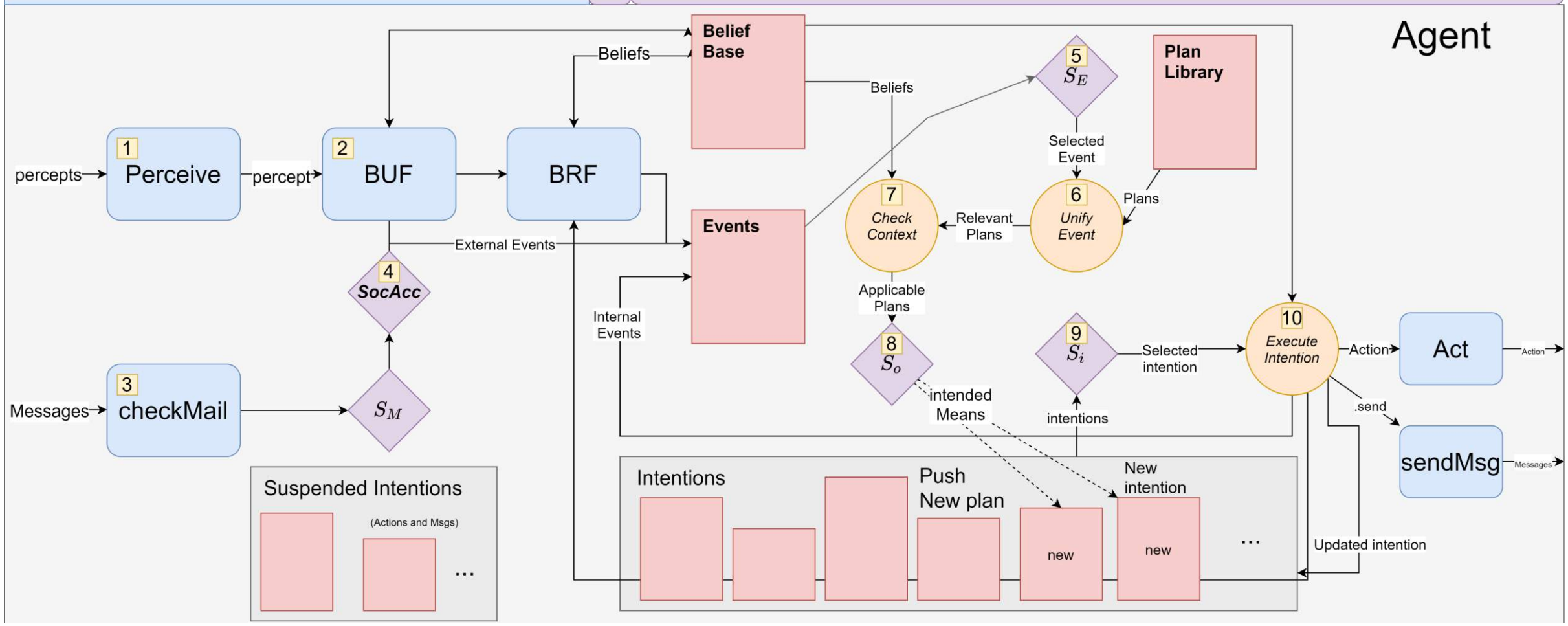
_____

The reasoning cycle

¿How the Jason interpreter runs an agent program?

An Agent operates by using a reasoning cycle, which can be divided into 10 main steps..

A	**Rectangles:** represent the main architectural components that determine the agent state.
B	**Circles:** represent functions used in reasoning cycle, these ones are essential and cannot be modified.
C	**Rounded boxes:** represent functions used in reasoning cycles, it can be customized by programmers.
D	**Diamonds:** represent functions used in reasoning cycle, it can be customized and are called selection function, because they take a list of items and the function selects one of them.

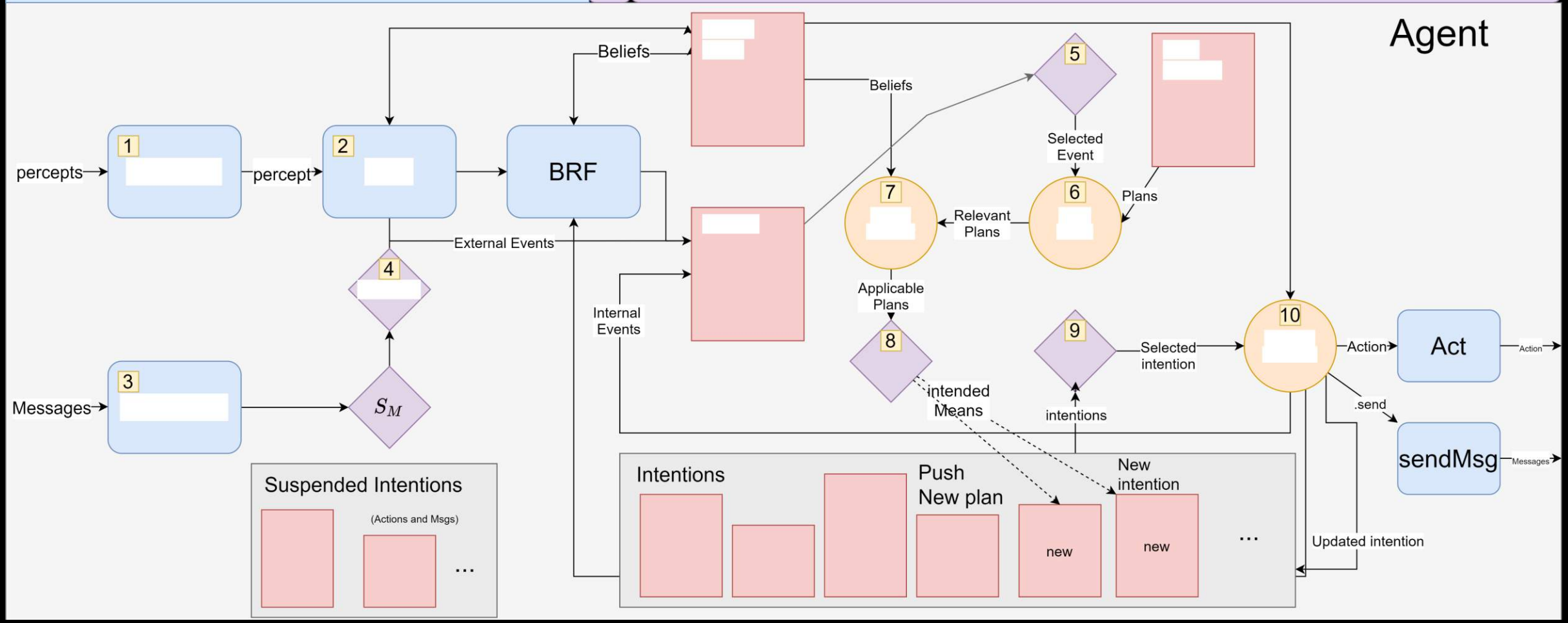


The reasoning cycle

¿How the Jason interpreter runs an agent program?

An Agent operates by using a reasoning cycle, which can be divided into main steps..

A	**Rectangles:** represent the main architectural components that determine the agent state.
B	**Circles:** represent functions used in reasoning cycle, these ones are essential and cannot be .
C	**Rounded boxes:** represent functions used in reasoning cycles, it can be by programmers.
D	**Diamonds:** represent functions used in reasoning cycle, it can be customized and are called selection function, because they take a list of and the function selects of them.



17 Reasoning cycle

Step 1 Perceiving the environment

The first thing an agent does in the **reasoning** cycle.

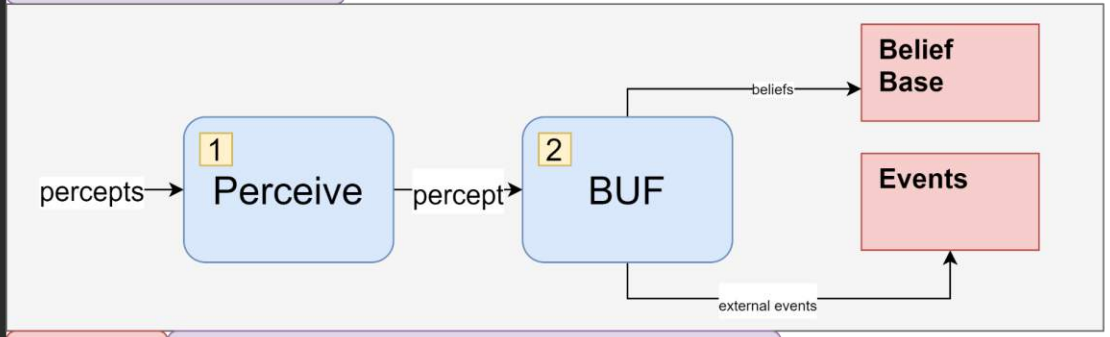
Is

Sense the environment.

This

Will be received in a symbolic form as a list of literals each literal is a **percept**.

Perceive method



Step 2 Updating the belief base

Once the list of percepts has been obtained, the belief base is updated to reflect the changes in the environments.

This is done by

The belief update function denoted as **BUF**

This method

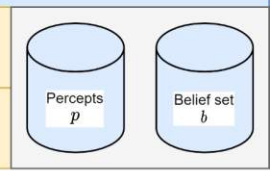
Assumes that every percept in the environment will be **included** in the lists of percepts.

17 Body

¿How does it work?

The BUF method updates the belief base in the following way.

A	Each literal $l \in p$ not in b is **added** to b.
B	Each literal $l \in b$ no longer in p is **removed** from b.



Events

Each change in the belief update function generates an event if this is resulting from perception of the environment is called **external event**.

Example

Consider an agent that perceives the color of a box.
The event generated will be:

<+Color(box1,red)[source(percept),T]>

If the next cycle the box is still perceived no event is generated.

However

If the agent does not perceive the box then the following event is generated

<-Color(box1,red)[source(percept),T]>

Both of the events will trigger a relevant **plan** to be executed.

17 Reasoning cycle

Step 1 Perceiving the environment

The first thing an agent does in the _____ cycle.

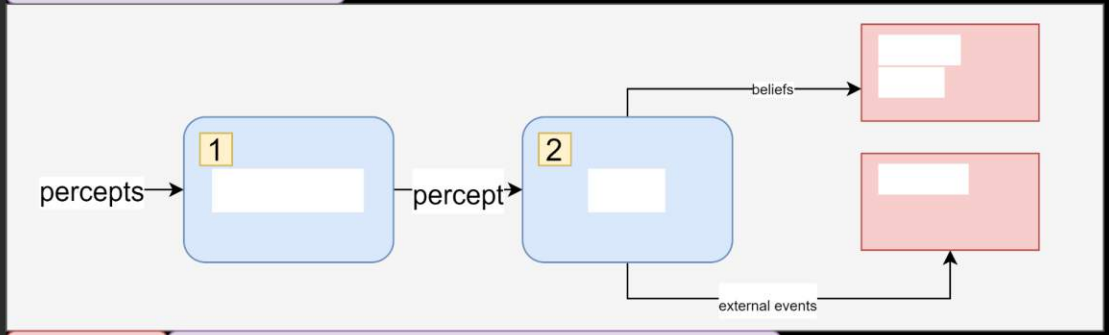
Is

_____ the environment.

This

Will be received in a symbolic form as a list of literals each literal is a _____.

Perceive method



Step 2 Updating the belief base

Once the list of percepts has been obtained, the belief base is updated to reflect the changes in the environments.

This is done by

The belief update function denoted as _____.

This method

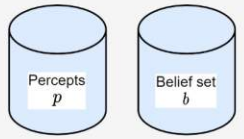
Assumes that every percept in the environment will be _____ in the lists of percepts.

17 Body

¿How does it work?

The BUF method updates the belief base in the following way.

A	Each literal $l \in p$ not in b is _____ to b.
B	Each literal $l \in b$ no longer in p is _____ from b.



Events

Each change in the belief update function generates an event if this is resulting from perception of the environment is called _____.

Example

Consider an agent that perceives the color of a box.
The event generated will be:

< _____ *[source(percept),T]*>

If the next cycle the box is still perceived no event is generated.

However

If the agent does not perceive the box then the following event is generated

<- _____ *[source(percept),T]*>

Both of the events will trigger a relevant _____ to be executed.

18 Step 3 - receiving communication from other agent

Introduction

One important source of information for an agent are other **agents**.

Method

This step of the reasoning cycle is check for messages from other agents in the agents **mailbox**.

This is done by

The ***checkMail*** method.

Processing the messages

In one reasoning cycle, only **one** message is processed, because of this agents will want to give priority to certain messages.

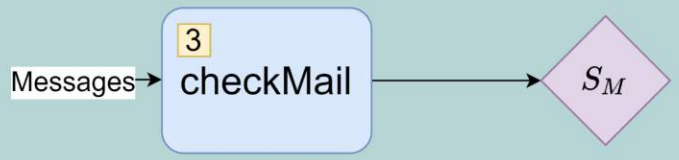
This is done by

The ***message selection function*** denoted as S_M

By default

Jason simply chooses the **first** message in the list.

Figure 18.1



19 Step 4 - Selecting "Socially acceptable" messages

Introduction

Before messages are processed the agent needs to determine if they can be accepted or not.

This is done by

Social acceptance function denoted as ***SocAcc***

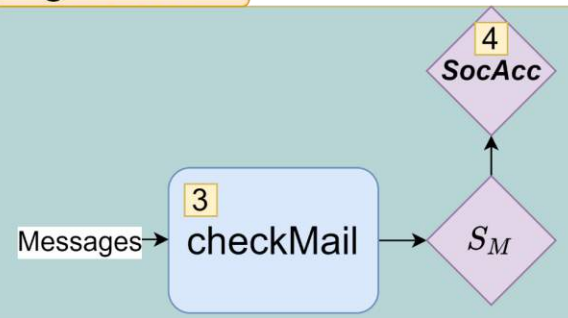
By default

This method accepts **all** messages from all agents.

You can think of this as

A spam filter

Figure 18.1



20 Step 5 - Selecting an event

Introduction

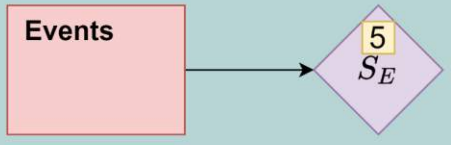
Agents operate by handling **events**, this are perceived changes in the environment or the agent's own goals.

Method

In each reasoning cycle only one pending event will be dealt with, therefore we need to select an event to be handled in a reasoning cycle, called ***selected event***.

This is done by

Event selection function denoted as S_E



By default

The function simply select the first ***event*** in the list of events

Finally

Each time an event is selected this will be removed from the set of events, if the set of events is empty, then the cycle proceeds directly to step ***9***.

18 Step 3 - receiving communication from other agent

Introduction

One important source of information for an agent are other _____.

Method

This step of the reasoning cycle is check for messages from other agents in the agents _____.

This is done by

The _____ method.

Processing the messages

In one reasoning cycle, only _____ message is processed, because of this agents will want to give priority to certain messages.

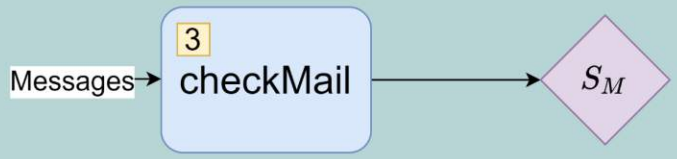
This is done by

The _____ denoted as S_M

By default

Jason simply chooses the _____ message in the list.

Figure 18.1



19 Step 4 - Selecting "Socially acceptable" messages

Introduction

Before messages are processed the agent needs to determine if they can be accepted or not.

This is done by

_____ denoted as _____

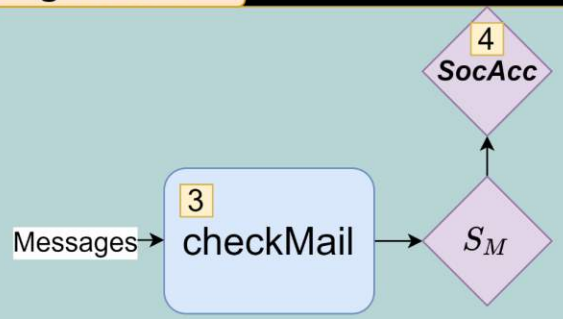
By default

This method accepts _____ messages from all agents.

You can think of this as

A spam filter

Figure 18.1



20 Step 5 - Selecting an event

Introduction

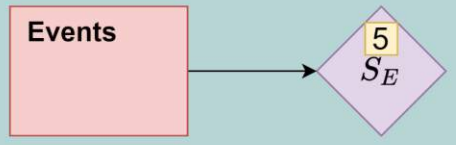
Agents operate by handling _____, this are perceived changes in the environment or the agent's own goals.

Method

In each reasoning cycle only one pending event will be dealt with, therefore we need to select an event to be handled in a reasoning cycle, called _____.

This is done by

_____ denoted as S_E



By default

The function simply select the first _____ in the list of events

Finally

Each time an event is selected this will be removed from the set of events, if the set of events is empty, then the cycle proceeds directly to step _____.

21 Step 6 - retrieving all relevant plans

Introduction

Now that there is a *selected event*, we need to find a plan that will allow the agent to handle that event.

Method

We do this finding in the plan library component all the plans which are relevant for the given event.

Consider this event

```
<+color(box1,blue)[source(percept),T]>
```

Consider this plans of the agents	Relevant
`@p1 +position(Object,color):...<-...`	No
`@p2 +color(Object,color):...<-...`	Yes
`@p3 +color(Object,color):...<-...`	Yes
`@p4 +color(Object,red):...<-...`	No
`@p5 +color(Object,color)[source(self)]:...<-..`	No
`@p6 +color(Object,color)[source(percept)]:.<-.`	Yes

Therefore

The set of relevant plans for this event is { **p2, p3, p6**}.

This is done by

The ***method unify event***

22 Step 7 - Determining the applicable plans

Introduction

We know plans have a context part which tell us whether a plan can be used at a particular moment in time.

For this step

We will need to select from the relevant plans, all those which are currently applicable.

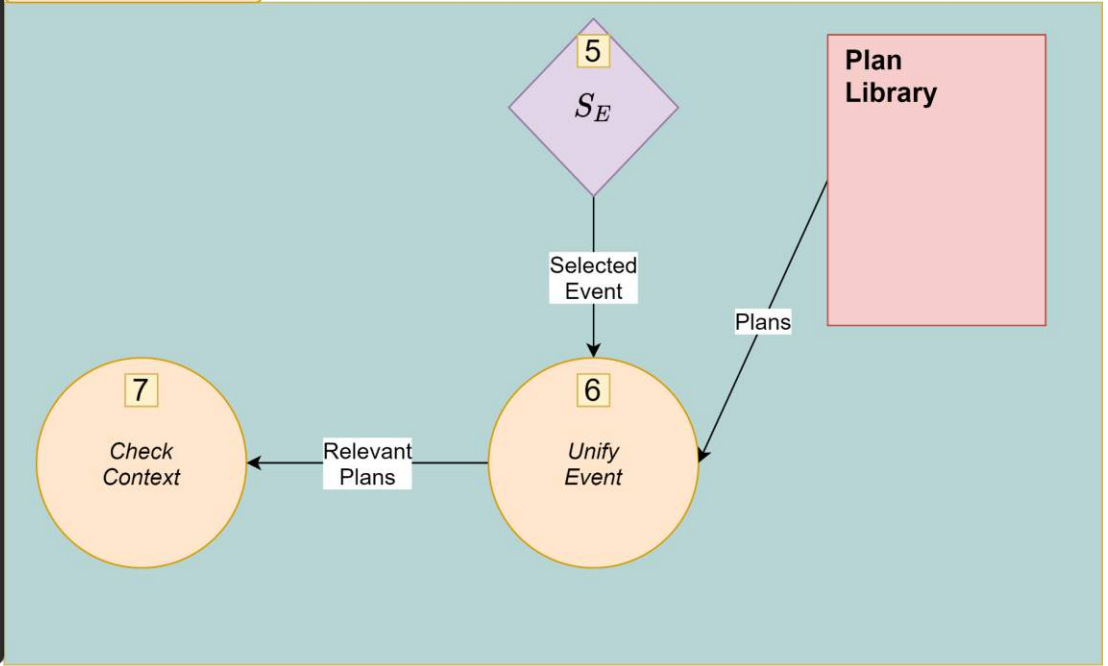
That means

Check whether the context of each of the relevant plans are **true**.

We will have

as a result a set of ***applicable plans***, example ***{p2,p6}***

Figure 18.1



21 Step 6 - retrieving all relevant plans

Introduction

Now that there is a _____, we need to find a plan that will allow the agent to handle that event.

Method

We do this finding in the plan library component all the plans which are relevant for the given event.

Consider this event

```
<+color(box1,blue)[source(percept),T]>
```

Consider this plans of the agents	Relevant
`@p1 +position(Object,color):...<-...`	
`@p2 +color(Object,color):...<-...`	
`@p3 +color(Object,color):...<-...`	
`@p4 +color(Object,red):...<-...`	
`@p5 +color(Object,color)[source(self)]:...<-..`	
`@p6 +color(Object,color)[source(percept)]:.<-.`	

Therefore

The set of relevant plans for this event is { _____ }.

This is done by

The _____

22 Step 7 - Determining the applicable plans

Introduction

We know plans have a context part which tell us whether a plan can be used at a particular moment in time.

For this step

We will need to select from the relevant plans, all those which are currently applicable.

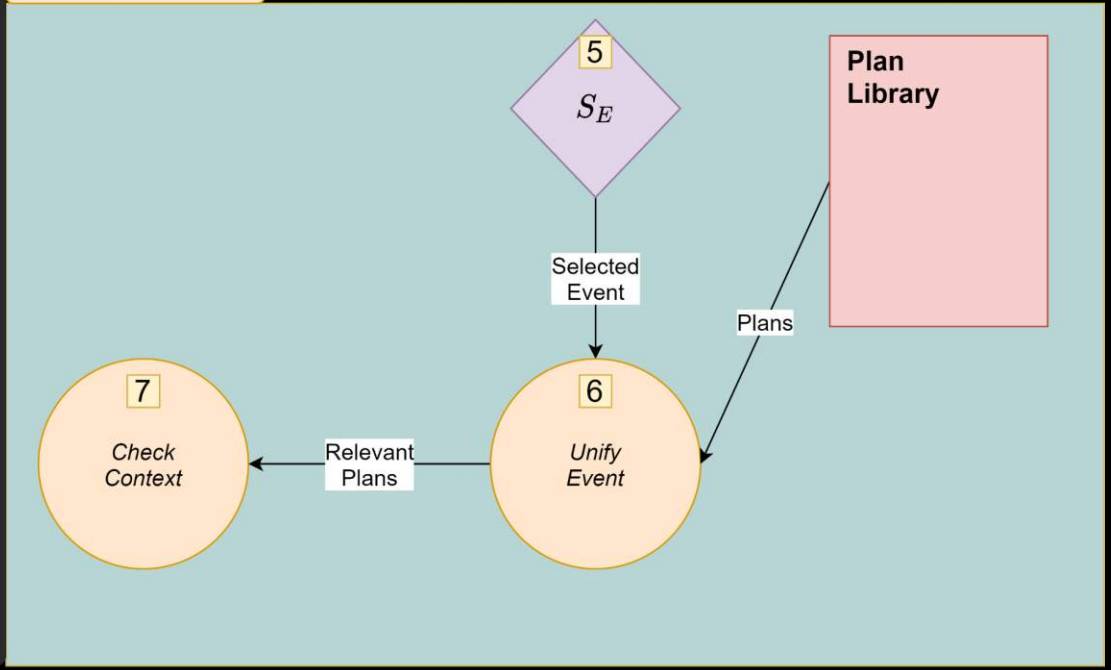
That means

Check whether the context of each of the relevant plans are _____.

We will have

as a result a set of _____, example _____

Figure 18.1



23 Step 8 - Selecting one applicable plan

Introduction

Now that we have our set of **applicable** plans, we can say that any of those plans are suitable of executing, these are considered as **options**.

Therefore

The agent needs to choose one of the applicable plans and **commit** to executing that particular plan.

Executing a plan

Commiting to a plan means that the agent has the **intention** of pursuing the course of action determined by that plan.

That means

Is added to the set of **intentions**

This is done by

The ***Option Selection function*** denoted by S_o

24 Step 9 - Selecting one applicable plan

Introduction

So far in the reasoning cycle we acquired a new intended means, generally an agent has more than one intention in the set of intentions.

Therefore

The next thing to do is choose one particular **intention** from out set of intentions.

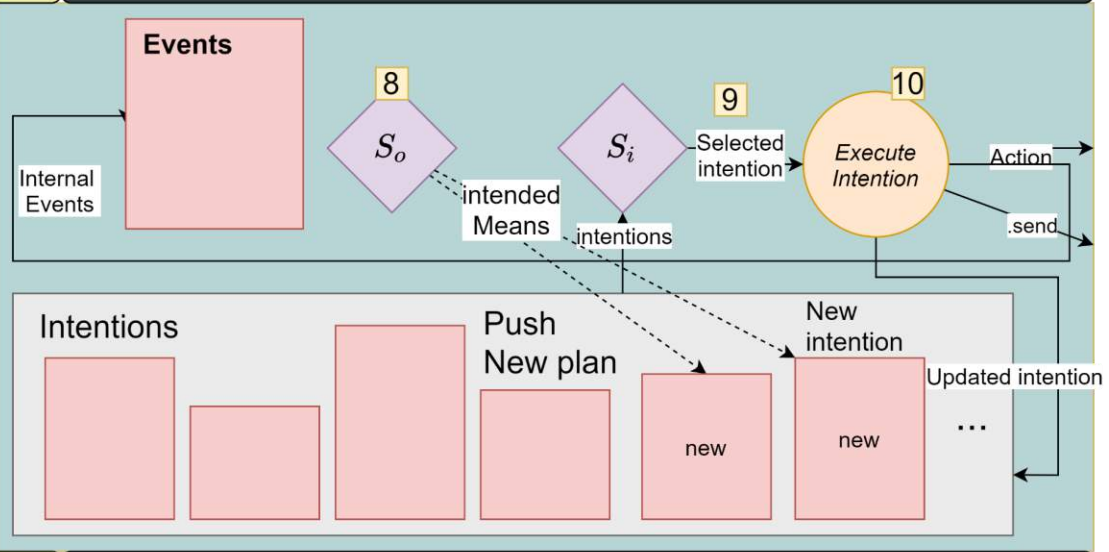
This is done by

Intention Selection function denoted by S_I

Finally

The intention will be removed from the list.

24 Step 8, 9 Figure



25 Step 10 - Executing one step of an intention

One thing the agent does in this step is to act upon the environment or follow up one of its intentions.

Body of a plan

We now that the body is a sequence of formulae and when a formulae is executed it is just deleted from the instantiated body plan.

26 Step 10 - Executing one step of an intention

Finally the cycle must clear intention that are completed.

This is

If the formulae executed in the current reasoning cycle was the **last** in the body of the plan, we eliminate that intention from the stack of intentions.

23 Step 8 - Selecting one applicable plan

Introduction

Now that we have our set of _____ plans, we can say that any of those plans are suitable of executing, these are considered as _____.

Therefore

The agent needs to choose one of the applicable plans and _____ to executing that particular plan.

Executing a plan

Commiting to a plan means that the agent has the _____ of pursuing the course of action determined by that plan.

That means

Is added to the set of _____.

This is done by

The _____ _____ _____ denoted by S_o

24 Step 9 - Selecting one applicable plan

Introduction

So far in the reasoning cycle we acquired a new intended means, generally an agent has more than one intention in the set of intentions.

Therefore

The next thing to do is choose one particular _____ from out set of intentions.

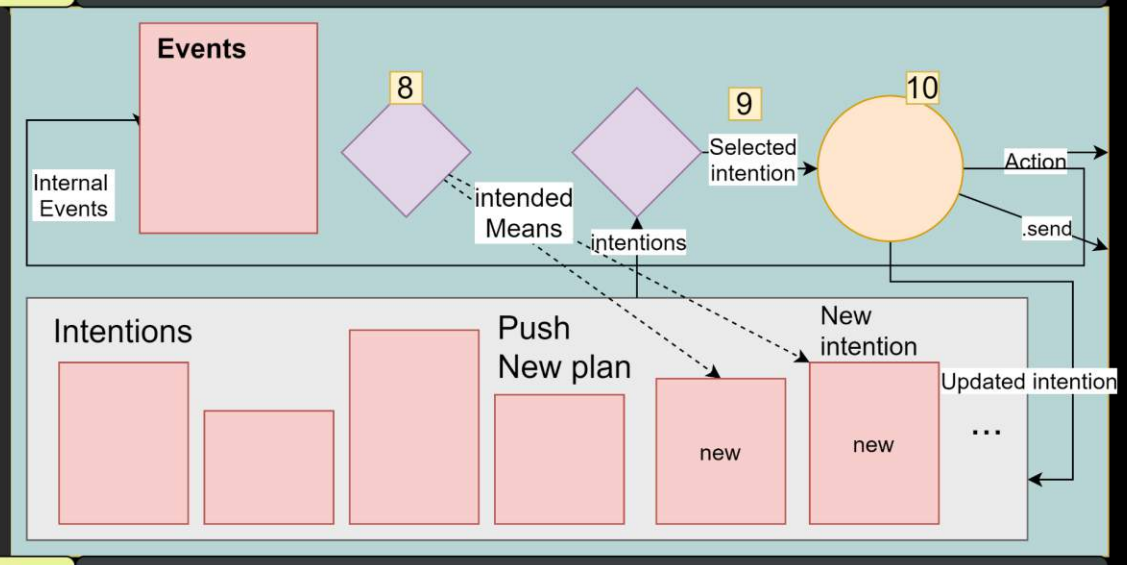
This is done by

_____ _____ _____ denoted by S_I

Finally

The intention will be removed from the list.

24 Step 8, 9 Figure



25 Step 10 - Executing one step of an intention

One thing the agent does in this step is to act upon the environment or follow up one of its intentions.

Body of a plan

We now that the body is a sequence of formulae and when a formulae is executed it is just deleted from the instantiated body plan.

26 Step 10 - Executing one step of an intention

Finally the cycle must clear intention that are completed.

This is

If the formulae executed in the current reasoning cycle was the _____ in the body of the plan, we eliminate that intention from the stack of intentions.

27 Plan failure

Introduction

It is common for a plan to **fail** to achieve the goal it was supposed to achieve.

Therefore

We must give an agent a plan for what to do when a preferred plan fails.

This are called

Contingency plans

We use

-!g triggering event to identify a contingency plan when ***+!g*** fails.

To specify that

backtracking should always be attempted, all we have to do is write the goal in a goal deletion plan.

28 Example of plan failure handling

```
@p1 +!g1     : true <- !g2(X); .print("end g1", X).
@p2 +!g2(X)  : true <- !g3(X); .print("end g2", X).
@p3 +!g3(X)  : true <- !g4(X); .print("end g3", X).
@p4 +!g4(X)  : true <- !g5(X); .print("end g4", X).
@p5 +!g5(X)  : true <- .fail.
@f1 -!g3(failure) : true <- .print("in g3 failure").
```

Here we have our initial goal g1, with only one relevant plan p1, this will create a sub goal and so the next one until it reaches g5 which will throw an error, this will trigger the contingency plan of g3 since its the only one relevant.

28 Example of plan failure handling

so the intention will look like this.

```
-!g3(failure)
                <- .print("in g3 failure")
+!g5(X)
                <- .fail.
+!g4(X)
                <- !g5(X); .print("end g4 ", X).
+!g3(X)
                <- !g4(X); .print("end g3 ", X).
+!g2(X)
                <- !g3(X); .print("end g2 ", X).
+!g1
                <- !g2(X); .print("end g1 ", X).
```

With resulting in...

```
[a] saying: in g3 failure
[a] saying: end g2 failure
[a] saying: end g1 failure
```

27 Plan failure

Introduction

It is common for a plan to to achieve the goal it was supposed to achieve.

Therefore

We must give an agent a plan for what to do when a preferred plan fails.

This are called

We use

triggering event to identify a contingency plan when ***+!g*** fails.

To specify that

backtracking should always be attempted, all we have to do is write the goal in a goal deletion plan.

28 Example of plan failure handling

```
@p1 +!g1     : true <- !g2(X); .print("end g1", X).
@p2 +!g2(X)  : true <- !g3(X); .print("end g2", X).
@p3 +!g3(X)  : true <- !g4(X); .print("end g3", X).
@p4 +!g4(X)  : true <- !g5(X); .print("end g4", X).
@p5 +!g5(X)  : true <- .fail.
@f1 -!g3(failure) : true <- .print("in g3 failure").
```

Here we have our initial goal g1, with only one relevant plan p1, this will create a sub goal and so the next one until it reaches g5 which will throw an error, this will trigger the contingency plan of g3 since its the only one relevant.

28 Example of plan failure handling

so the intention will look like this.

```
-!g3(failure)
                <- .print("in g3 failure")
+!g5(X)
                <- .fail.
+!g4(X)
                <- !g5(X); .print("end g4 ", X).
+!g3(X)
                <- !g4(X); .print("end g3 ", X).
+!g2(X)
                <- !g3(X); .print("end g2 ", X).
+!g1
                <- !g2(X); .print("end g1 ", X).
```

With resulting in...

```
[a] saying: in g3 failure
[a] saying: end g2 failure
[a] saying: end g1 failure
```

29 Interpreter configuration and execution modes

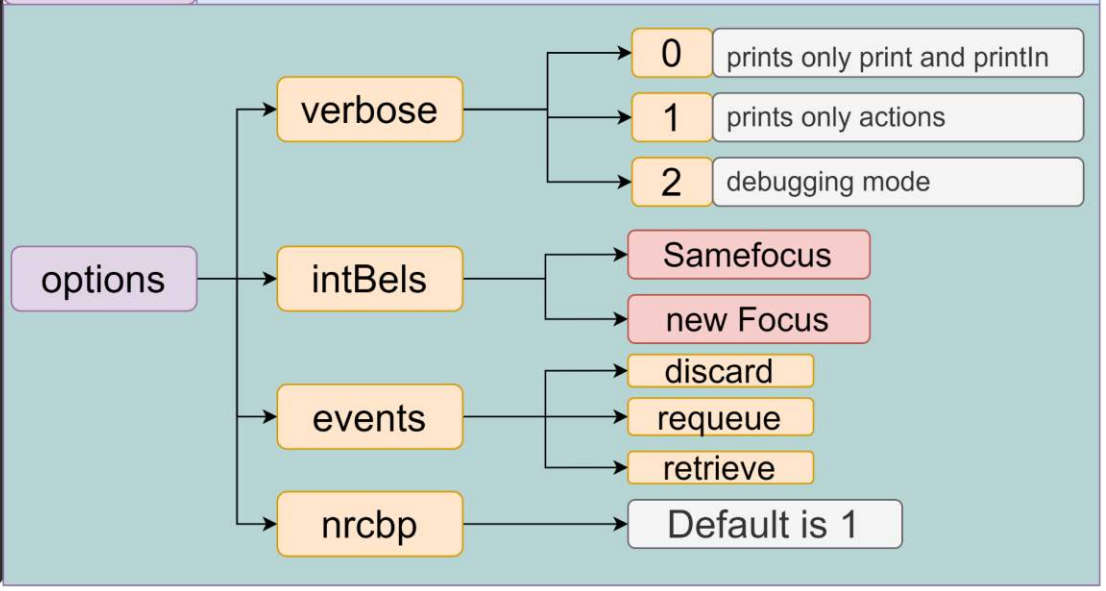
As we know jason uses a project file where multi-agent system is defined

General form of definition

```
MAS <name>  {
infrastructure  : <Centralised|Saci|Jade|...>
environment     : <environment_class> at <host>
executionControl: <execution_class> at <host>

agents          : <ag1_name><file>  <options>
                   agentArchClass   <arch_class>
                   agentClass       <ag_class>
                   beliefBaseClass  <bb_class>
                   # <num_instances> at <host> ;
                   <ag1_name>...;
}
```

Options for the options we have



29 Interpreter configuration and execution modes

User settings: users can create their own settings in the agent declaration

Example

```
Agents: ag1[Verbose = 2, file="an.xml", value=45]
```

This parameters are stored in the setting class and can be accesed by

```
getSettings().getUserParameter("file");
```

Execution modes

We can also change the execution modes there are three different execution modes.

1 Asynchronous

all agents run asynchronously.
An agent goes to its next reasoning cycle as soon as it has finished its current cycle. This is the default execution mode

2 Synchronous

all agents perform one reasoning cycle at every ‘global execution step’.
When an agent finishes its waits for the controller to a ‘carry on’ signal.

```
executionControl: jason.control.ExecutionControl
```

3 Debugging

Similar to synchronous, the difference is that it will also wait for the user to click the "step" button.

```
executionControl: jason.control.ExecutionControlGUI
```

29 Interpreter configuration and execution modes

As we know jason uses a project file where multi-agent system is defined

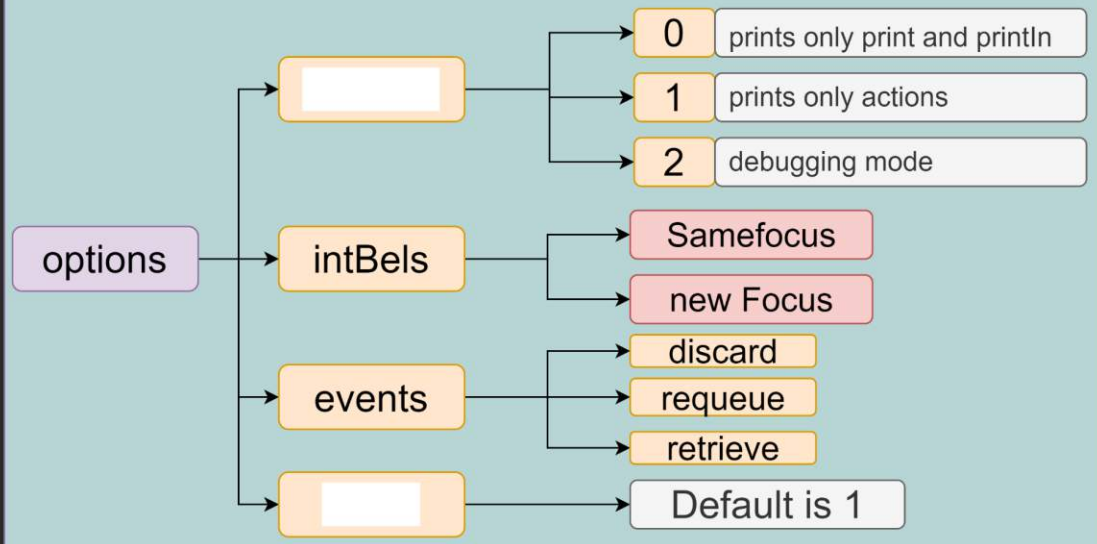
General form of definition

```
MAS <name>  {
infrastructure  : <Centralised|Saci|Jade|...>
environment     : <environment_class> at <host>
executionControl: <                  > at <host>

agents          : <ag1_name><file>  <        >
                   agentArchClass   <arch_class>
                   agentClass       <ag_class>
                   beliefBaseClass  <bb_class>
                   # <num_instances> at <host> ;
                   <ag1_name>...;
}
```

Options

for the options we have



29 Interpreter configuration and execution modes

User settings: users can create their own settings in the agent declaration

Example

```
Agents: ag1[        = 2, file="an.xml", value=45]
```

This parameters are stored in the setting class and can be accesed by

```
getSettings().                  ("file");
```

Execution modes

We can also change the execution modes there are three different execution modes.

1 Asynchronous

all agents run asynchronously.
An agent goes to its next reasoning cycle as soon as it has finished its current cycle. This is the default execution mode

2 Synchronous

all agents perform one reasoning cycle at every ‘global execution step’.
When an agent finishes its waits for the controller to a ‘carry on’ signal.

```
executionControl: jason.control.
```

3 Debugging

Similar to synchronous, the difference is that it will also wait for the user to click the "step" button.

```
executionControl: jason.control.
```

30 Predefined plan annotations

Plans can have associate information about the plans, so that the user can write a selection function that access such information to facilitate the task.

This are

```
@Plan[atomic, breakpoint, all_unifs, priority]
```

1 Atomic

This plan or intention will be executed until it **finishes.**

2 breakpoint

If debug mode is being used, the execution stops and the control goes back to the user.

3 all_unifs

Is used to choose between different plans.

4 Priority

Is a term of arity 1, it gives a priority, the higher the number, the higher the priority order.

31 Excercices

Consider the following plan library

```
@p1 +g(X,Y) : true                                  <- ... .
@p2 +g(X,Y) : a(Y) & not b(X)                       <- ... .
@p3 +g(X,_) : a(Y) & Y > x                          <- ... .
@p4 +g(X,Y)[source(self)] : true                    <- ... .
@p5 +g(X,Y)[source(self), source(ag1)] : true <- ... .
@p6[all_unifs] : a(Y) & Y > x                       <- ... .
```

And the following beliefs

```
a(10).
a(5).
b(20)
```

If the next event is selected

```
<+g(10,5)[source(Ag1)]>
```

Which plans are relevant and which are applicable

Relevant	{1,2,5,6}
Applicable	{5,1}

30 Predefined plan annotations

Plans can have associate information about the plans, so that the user can write a selection function that access such information to facilitate the task.

This are

```
@Plan[atomic, breakpoint, all_unifs, priority]
```

1

This plan or intention will be executed until it _____.

2

If debug mode is being used, the execution stops and the control goes back to the user.

3

Is used to choose between different plans.

4

Is a term of arity 1, it gives a priority, the higher the number, the higher the priority order.

31 Excercices

Consider the following plan library

```
@p1 +g(X,Y) : true                                  <- ... .
@p2 +g(X,Y) : a(Y) & not b(X)                       <- ... .
@p3 +g(X,_) : a(Y) & Y > x                          <- ... .
@p4 +g(X,Y)[source(self)] : true                    <- ... .
@p5 +g(X,Y)[source(self), source(ag1)] : true <- ... .
@p6[all_unifs] : a(Y) & Y > x                       <- ... .
```

And the following beliefs

```
a(10).
a(5).
b(20)
```

If the next event is selected

```
<+g(10,5)[source(Ag1)]>
```

Which plans are relevant and which are applicable

Relevant	
Applicable	

32 Environments

Introduction

Agents are situated in an environment and this is shared by multiple agents, this means agent`s actions can interfere with those other agents, therefore we need a good notion of an **environment**.

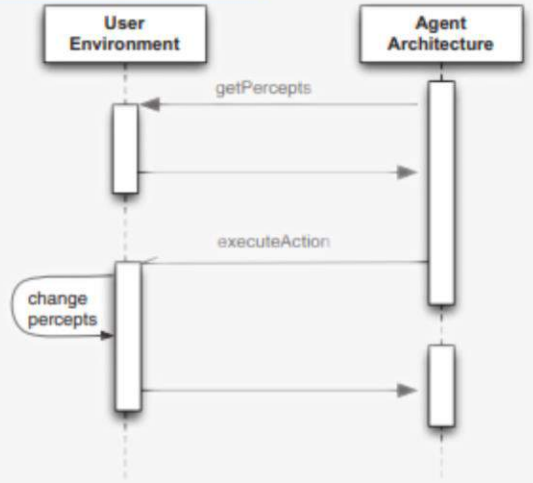
For this

We need to create a computational model of a real world or artificial environment, and we need to simulate the dynamic aspects of that environment.

How we do this

The arquitecture of an agent include java methods which implement the interaction of an agent with its environment.

Figure 32.1

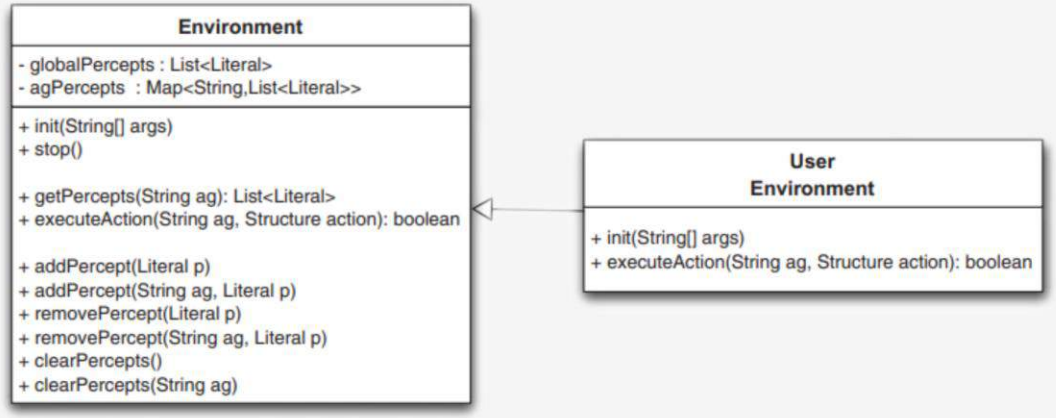


We see that the agent arquitecture use **getPercepts** to retrieve percepts of the environment

33 The environment class

In order to implement an environment in jason, programmers need to extend the environment class, and likely override **executeAction** an **init**.

Figure 33.1



globalPercepts and agPercepts

The environment class keeps data **structures** that are used to store percepts to which the agent has access as well as 'global' percepts that all agents perceive.

getPercepts

The method get percepts checks the **percepts** available to the particular agent calling the method

32 Environments

Introduction

Agents are situated in an environment and this is shared by multiple agents, this means agent`s actions can interfere with those other agents, therefore we need a good notion of an _____.

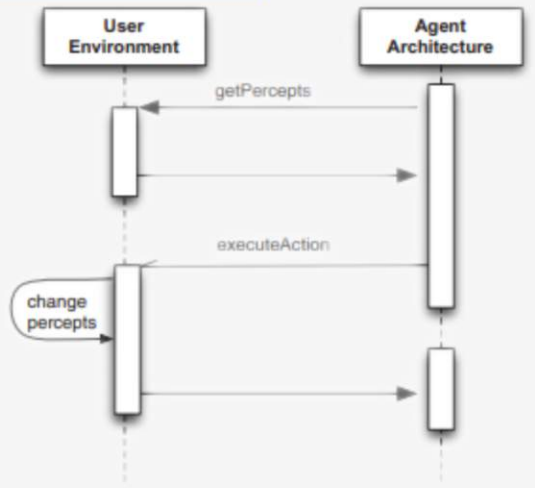
For this

We need to create a computational model of a real world or artificial environment, and we need to simulate the dynamic aspects of that environment.

How we do this

The arquitecture of an agent include java methods which implement the interaction of an agent with its environment.

Figure 32.1

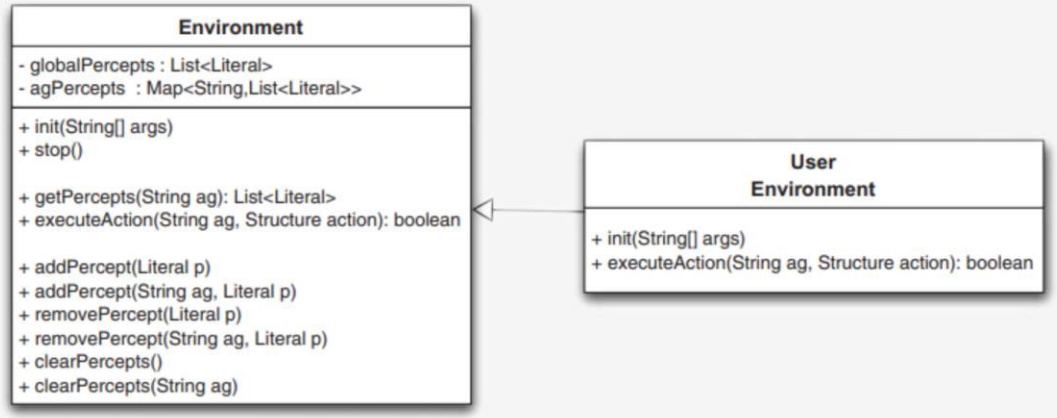


We see that the agent arquitecture use _____ to retrieve percepts of the environment

33 The environment class

In order to implement an environment in jason, programmers need to extend the environment class, and likely override _____ an ____.

Figure 33.1



globalPercepts and agPercepts

The environment class keeps data _____ that are used to store percepts to which the agent has access as well as 'global' percepts that all agents perceive.

getPercepts

The method get percepts checks the _____ available to the particular agent calling the method

34 Methods

The init method

it can be use to receive parameters from the environment class that is the system configuration(**.masj**)

It is appropiate

To use this method to initialize the lists of **percepts**.

The parse literal method

Percepts are typically created using the parse literal method of the **literal** class

Methods used in jason environment

1 addPercept(L)

add literal L to the global list of percepts; that is, all agents will perceive L if they sense the environment before L is removed from the global percepts

2 addPercept(A,L)

add literal L to the list of percepts that are exclusive to agent A; that is, only A will be able to perceive L;

3 removePercept(L)

add literal L to the list of percepts that are exclusive to agent A; that is, only A will be able to perceive L;

4 removePercept(A,L)

remove literal L from the list of percepts that
are exclusive to agent A;

5 clearPercepts() || clearPercepts(A)

: delete all percepts in the list of percepts that are
exclusive to agent A or global

35 The environment class

Only instances of the class literal are added to the lists of percepts.

The execute action method

Every time an agent tries to execute an environment action, the name of the agent and a structure representing the action are sent as **parameters** to this method.

We should note

The execution of an actions needs to return a Boolean value, whether the agent's request for an action execution was attempted or refused by the environment.

Errors of programmers

There are two errors made by beginners when programing multi-agent systems

1 They expect the agent to keep the perceived information

If an agent needs to remember, there need to be **mental notes**, mental notes are remembered until explicitly deleted.

2 Mismatches between percepts and actions names

```
import jason.asSyntax.*;
import jason.environment.*;
public class extends Environment
{
    @Override
    public void init(String[] args){
        // setting initial (global) percepts ...
    }
    @Override
    public boolean executeAction(String ag, Structure act)
    {
    }
}
```

34 Methods

The init method

it can be use to receive parameters from the environment class that is the system configuration(.)

It is appropiate

To use this method to initialize the lists of .

The parse literal method

Percepts are typically created using the parse literal method of the class

Methods used in jason environment

1 (L)

add literal L to the global list of percepts; that is, all agents will perceive L if they sense the environment before L is removed from the global percepts

2 (A,L)

add literal L to the list of percepts that are exclusive to agent A; that is, only A will be able to perceive L;

3 (L)

add literal L to the list of percepts that are exclusive to agent A; that is, only A will be able to perceive L;

4 (A,L)

remove literal L from the list of percepts that
are exclusive to agent A;

5 () || (A)

: delete all percepts in the list of percepts that are
exclusive to agent A or global

35 The environment class

Only instances of the class literal are added to the lists of percepts.

The execute action method

Every time an agent tries to execute an environment action, the name of the agent and a structure representing the action are sent as to this method.

We should note

The execution of an actions needs to return a Boolean value, whether the agent's request for an action execution was attempted or refused by the environment.

Errors of programmers

There are two errors made by beginners when programing multi-agent systems

1 They expect the agent to keep the perceived information

If an agent needs to remember, there need to be , mental notes are remembered until explicitly deleted.

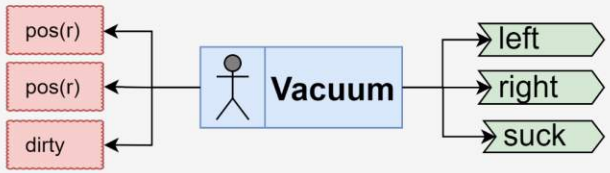
2 Mismatches between percepts and actions names

```
import jason.asSyntax.*;
import jason.environment.*;
public class extends
{
    @Override
    public void (String[] args){
        // setting initial (global) percepts ...
    }
    @Override
    public boolean (String ag, Structure act)
    {
    }
}
```

36 The environment class

Lets consider a concrete example of an environment

Prometheus notation



Codigo

```
public class VCWorld extends Environment
{
    boolean[] dirty = {true, true};

    char VacuumPos = 0;

    Random r = new Random();
}
```

We set the position of our vacuum in 0, since we only have two position we use an array of dirty.

```
@Override
public void init(String[] args)
{
    UpdatePercepts();
}
```

We override our init method and update our percepts.

37 The environment class

We set the dirtiness, and clear global percepts.

```
private void UpdatePercepts()
{
    if(r.nextInt(100)< 20)
    {
       dirty[r.nextInt(2)] = true;
    }
    clearPercepts();
    ...
}
```

Now we add the percepts if our vacuum is in a dirty place.

```
private void UpdatePercepts()
{
    ...
    if(dirty[vacuumPos])
        addPercept(Literal.parseLiteral("dirty"));
    else
        addPercept(Literal.parseLiteral("clean"));
```

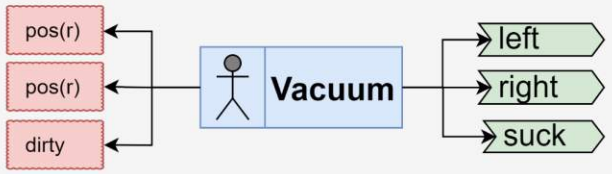
```
    if(vacuumPos == 0)
        addPercept(Literal.parseLiteral("pos(1)"));
    else
        addPercept(Literal.parseLiteral("pos(r)"));
}
```

We create the percept of the position of the vacuum in the environment.

36 The environment class

Lets consider a concrete example of an environment

Prometheus notation



Codigo

```
public class VCWorld extends Environment
{
    boolean[] dirty = {true, true};

    char VacuumPos = 0;

    Random r = new Random();
}
```

We set the position of our vacuum in 0, since we only have two position we use an array of dirty.

```
@Override
public void init(String[] args)
{
                    ();
}
```

We override our init method and update our percepts.

37 The environment class

We set the dirtiness, and clear global percepts.

```
private void UpdatePercepts()
{
    if(r.nextInt(100)< 20)
    {
       dirty[r.nextInt(2)] = true;
    }
                    ();
    ...
}
```

Now we add the percepts if our vacuum is in a dirty place.

```
private void UpdatePercepts()
{
    ...
    if(dirty[vacuumPos])
        addPercept(Literal.parseLiteral("        "));
    else
        addPercept(Literal.parseLiteral("        "));
```

```
   if(vacuumPos == 0)
        addPercept(Literal.parseLiteral("        "));
    else
        addPercept(Literal.parseLiteral("        "));
}
```

We create the percept of the position of the vacuum in the environment.

EXCERCICES AGENTSPEAK

1 Exercises 1

1 First exercise

Bob: a nice man.

Objective

Bob is happy so when somebody greets him he salutes back

Description

Bob has the property ***happy***, bob has the desire to ***say(hello)***, bob has a plan to do this, so whenever bob has the desire to ***say(X)***, and he believes he is ***happy(bob)***, by executing ***print(X)***, he will achieve this desire.

Code

```
happy(bob).

!say(hello).

+!say(X) : happy(bob) <- .print(X).
```

Output

```
MAS Console - mentalist
Jason Http Server running on http://192.168.1.3:3272
[bob] hello
```

1 Exercises 2

2 Extend first exercise

Bob: a nice man who happens to be a polyglot.

Objective

Bob is happy, so when somebody greets him he salutes back, but he can do it in three languages.

Description

Bob has the property ***happy***, he also has the properties msg, which have their translation of hello in english, french and pt.
bob has the desire to ***say(pt|fr|en)***.
bob has a plan to do this
so whenever he have de desire to ***say(Y),*** and he believes the msg(Y is X) and believes he is happy. by executing .print(X), he will achieve this desire.

Code

```
happy(bob).
msg(en,"hello").
msg(fr,"Bonjour").
msg(pt,"Ola").

!say(pt).

+!say(Y) : msg(Y,X) & happy(bob) <- .print(X).
```

Output

```
MAS Console - mentalist
Jason Http Server running on http://192.168.1.3:3272
[bob] hello
```

Exercises 1

1 First exercise

Bob: a nice man.

Objective

Bob is happy so when somebody greets him he salutes back

Description

Bob has the property ***happy***, bob has the desire to ***say(hello)***, bob has a plan to do this, so whenever bob has the desire to ***say(X)***, and he believes he is ***happy(bob)***, by executing ***print(X)***, he will achieve this desire.

Code

Output

MAS Console - mentalist

```
Jason Http Server running on http://192.168.1.3:3272
[bob] hello
```


Exercises 2

2 Extend first exercise

Bob: a nice man who happens to be a polyglot.

Objective

Bob is happy, so when somebody greets him he salutes back, but he can do it in three languages.

Description

Bob has the property ***happy***, he also has the properties msg, which have their translation of hello in english, french and pt.
bob has the desire to ***say(pt|fr|en)***.
bob has a plan to do this
so whenever he have de desire to ***say(Y),*** and he believes the msg(Y is X) and believes he is happy. by executing .print(X), he will achieve this desire.

Code

Output

MAS Console - mentalist

```
Jason Http Server running on http://192.168.1.3:3272
[bob] hello
```

1 Exercises 3

1 First exercise

Bob and alice

Objective

Alice tell bob that he is happy

bob

whenever **bob** has the belief that he is ***happy(bob)***, he will have the desire to ***say(hello)***, **bob** has a plan to do this whenever he comes to have the desire to ***say(X)*** and he believes that today is not monday, by ***.print(X)***, and then, ***.wait(500)***, follow by having the desire to ***say(X)***, he will achieve this.

alice

alice has the property ***started***, whenever **alice** has come to believe ***started***, she will tell ***bob***, that he is ***happy*** with ***.send()***.

Code

```
+happy(bob) <- !say(hello).

+!say(X): not today(Monday)
        <- .print(X);
           .wait(500);
           .!say(X);
```

Code

```
started.

+started <-
  .send(bob,tell,happy(bob));
```

Output

```
Jason Http Server running on http://192.168.1.3:3272
[bob] hello
[bob] hello
[bob] hello
[bob] hello
[bob] hello
```

1 Exercises 4

2 Extend first exercise

Bob, Alice and marcos, but alice is sincere and marcos is not

Objective

both Alice and marcos tell bob that he is happy, but bob only believes alice

Description

bob has the property ***sincere(alice)***, whenever **bob** has come to believe ***happy(bob)*** *from any source*, and he believes according to himself that source is **sincere(A)** , he have the desire to ***say(hello(A))*** to that source, whenever he comes to have the desire to ***say(x)***, and he believe is not **Monday**, he will achieve this by ***print(x)***, ***.wait(500)***, and having the desire to ***say(x)*** again.

alice and marcos

alice has the property ***started***, whenever **alice** has come to believe ***started***, she will tell ***bob***, that he is ***happy*** with ***.send()***.

Code

```
sincere(alice).
+happy(bob)[source(A)] :
sincere(A)[source(self)]
        <- !say(hello(A)).
+!say(X) : not today(monday)
        <- .print(X);
           .wait(500);
            !say(X).
```

```
started.

+started <-

.send(bob,tell,happy(bob));
```

Output

```
Jason Http Server running on http://192.168.1.3:3272
[alice] done
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
```

1

Exercises 3

1 First exercise

Bob and alice

Objective

Alice tell bob that he is happy

bob

whenever **bob** has the belief that he is ***happy(bob)***, he will have the desire to ***say(hello)***, **bob** has a plan to do this whenever he comes to have the desire to ***say(X)*** and he believes that today is not monday, by ***.print(X)***, and then, ***.wait(500)***, follow by having the desire to ***say(X)***, he will achieve this.

alice

alice has the property ***started***, whenever **alice** has come to believe ***started***, she will tell ***bob***, that he is ***happy*** with ***.send()***.

Code

Code

Output

```
Jason Http Server running on http://192.168.1.3:3272
[bob] hello
[bob] hello
[bob] hello
[bob] hello
[bob] hello
```

1

Exercises 4

2 Extend first exercise

Bob, Alice and marcos, but alice is sincere and marcos is not

Objective

both Alice and marcos tell bob that he is happy, but bob only believes alice

Description

bob has the property ***sincere(alice)***, whenever **bob** has come to believe ***happy(bob)*** *from any source*, and he believes according to himself that source is **sincere(A)** , he have the desire to ***say(hello(A))*** to that source, whenever he comes to have the desire to ***say(x)***, and he believe is not **Monday**, he will achieve this by ***print(x)***, ***.wait(500)***, and having the desire to ***say(x)*** again.

alice and marcos

alice has the property ***started***, whenever **alice** has come to believe ***started***, she will tell ***bob***, that he is ***happy*** with ***.send()***.

Code

Output

```
Jason Http Server running on http://192.168.1.3:3272
[alice] done
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
[bob] hello(alice)
```

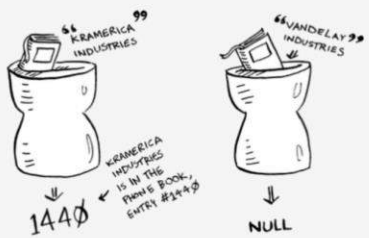
ALGORITMOS

Introduction

Suppose you are searching for someone in a phone book, their name starts with ***k***.
you could start at the beginning, but you are more likely to start at a page in the middle, because you know ***k*** is going to be near the middle.

¿What is it?

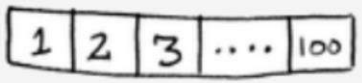
Binary search is an algorithm, that takes as input a ***sorted*** list of elements, and as an output it will returns the ***position*** of the element you are looking for

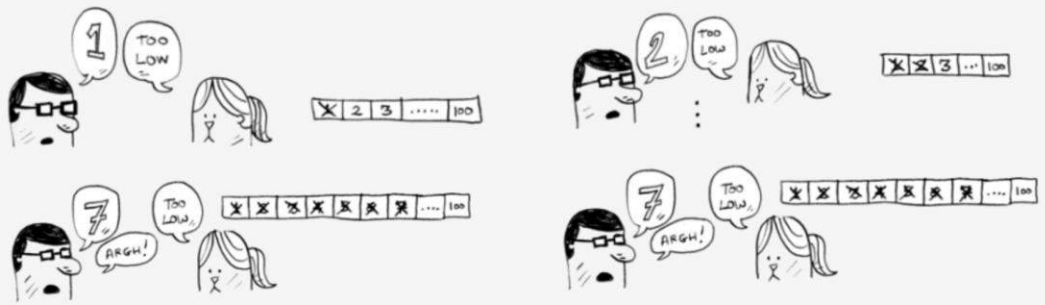


Bad example

Simple Search

Try to guess a number between 0 to 100





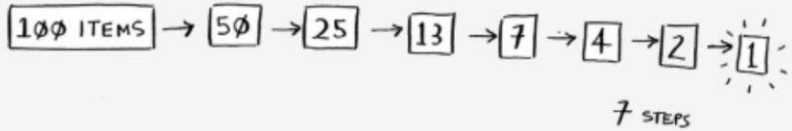
Just removing one number at a time, in the worst case we will need ***99*** steps!

1 Binary search By Aditya Y. Bhargava

Good example

Too low, but you just eliminated ***half*** the numbers!

This is binary search, in this algorithm you eliminate ***half*** of the guesses at each ***iteration***.



Exercises

Suppose you are looking for a word in the dictionary. The dictionary has 240,000 words, in the worst case, ¿how many steps do you think each search will take?

Simple search	***18***	steps
Binary search	***240,000***	steps

Introduction

Suppose you are searching for someone in a phone book, their name starts with ***k***.
you could start at the beginning, but you are more likely to start at a page in the middle, because you know ***k*** is going to be near the middle.

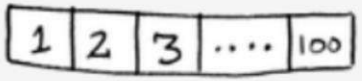
¿What is it?

Binary search is an algorithm, that takes as input a _____ list of elements, and as an output it will returns the _____ of the element you are looking for

Bad example

Simple Search

Try to guess a number between 0 to 100



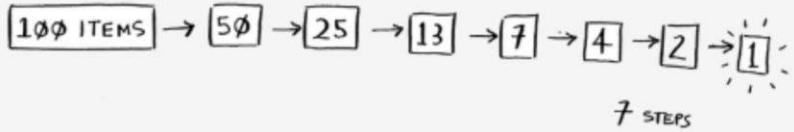
Just removing one number at a time, in the worst case we will need _____ steps!

1 Binary search By Aditya Y. Bhargava

Good example

Too low, but you just eliminated _____ the numbers!

This is binary search, in this algorithm you eliminate _____ of the guesses at each _____.



Exercises

Suppose you are looking for a word in the dictionary. The dictionary has 240,000 words, in the worst case, ¿how many steps do you think each search will take?

Simple search		steps
Binary search		steps

In general for any list of ***n***, binary search will take $log_2 n$ in the worst case, whereas simple search will take ***n*** steps.

Logarithms

You may not remember what logarithms are, but you probably know what exponentials are.
$log_{10} 100$ is like asking, **"How many 10s do we multiply together to get 100?"**

Exercises

$10^2 = 100$	$log_{10} 100 = 2$
$10^3 = 1000$	$log_{10} 1000 = 3$
$2^3 = 8$	$log_2 8 = 3$
$2^4 = 16$	$log_2 16 = 4$
$2^5 = 32$	$log_2 32 = 5$

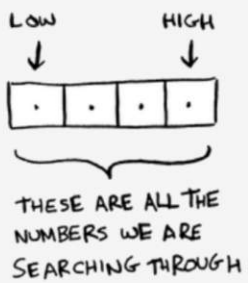
Note

Binary search only works when your list is in ***sorted*** order.

Binary search algorithm

takes a sorted array and an item, if the item is in the array, the function returns its position.

```
Low  = 0;
High = list.Lenght() - 1;
```



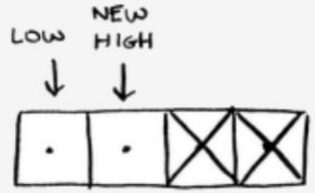
2 Binary search By Aditya Y. Bhargava

```
mid   = (low + high) / 2;
guess = list[mid]
```

mid is rounded down if the number is not even

If the guess is too low, you update low accordingly.

```
if guess < item;
    low = mid + 1;
```



Algorithm

```
function binary search(list, item)
    low = 0;
    high = list.length() - 1;
    while (low <= high)
        mid = (low + high)/2;
        guess = list[mid];
        if(guess == item)
            return mid;
        if(guess > item)
            high = mid - 1;
        else{low = mid + 1}
    return null;
```

- Low and high keep track of which part of the list you are searching.
- while you havent narrowed it to one element.
- check the middle element.
- found the item!.
- it was to high.
- it was to low
- the item does not exists

Exercise

Suppose you have a sorted list of 128 names, what is the maximum number of steps it would take?

Suppose you doubled the size of the list. What is the maximum number of steps now?

In general for any list of ***n***, binary search will take _____ in the worst case, whereas simple search will take ____ steps.

Logarithms

You may not remember what logarithms are, but you probably know what exponentials are.
$log_{10}100$ is like asking, **"How many _____ do we multiply together to get 100?"**

Exercises

	↔	$log_{10}100 = 2$
$10^3 = 1000$	↔	
	↔	$log_2 8 = 3$
$2^4 = 16$	↔	
	↔	$log_2 32 = 5$

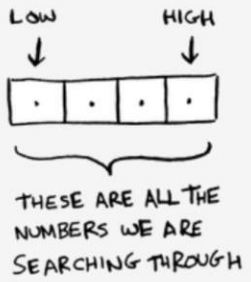
Note

Binary search only works when your list is in _____ order.

Binary search algorithm

takes a sorted array and an item, if the item is in the array, the function returns its position.

```
Low  = 0;
High = list.Lenght() - 1;
```



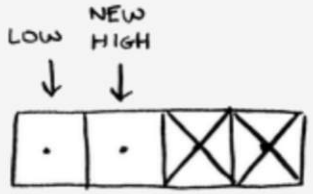
2 Binary search By Aditya Y. Bhargava

```
mid   = (low + high) / 2;
guess = list[mid]
```

mid is rounded down if the number is not even

If the guess is too low, you update low accordingly.

```
if guess < item;
    low = mid + 1;
```



Algorithm

```
function binary search(list, item)
    low = 0;
    high = list.length() - 1;
    while (low <= high)
        mid = (low + high)/2;
        guess = list[mid];
        if(guess == item)
            return mid;
        if(guess > item)
            high = mid - 1;
        else{low = mid + 1}
    return null;
```

Exercise

Suppose you have a sorted list of 128 names, what is the maximum number of steps it would take?

Suppose you doubled the size of the list. What is the maximum number of steps now?

Generally you want to use the most efficient algorithm.
back with the simple search we check each number, if this is a list of 100, it takes up to 100 guesses, if this is a list of 4 billion, then it takes 4 billion guesses, so the maximum number of guesses **is the same as the size of the list.**

Linear time

An algorithm is said to take linear time when the running time increases at most linearly with the size of the input, this means, the maximum time is the same size as the size of the input.

Running time binary search

Binary search is different if the list is 100 items long, it takes at most 7 guesses, if the list is 4 billion items, it takes at most 32 guesses.

Logarithmic time of log time

An algorithm is said to take logarithmic time when the running time grows in proportion to the logarithm of the input size.

2 Binary search By Aditya Y. Bhargava

Figure 2.1

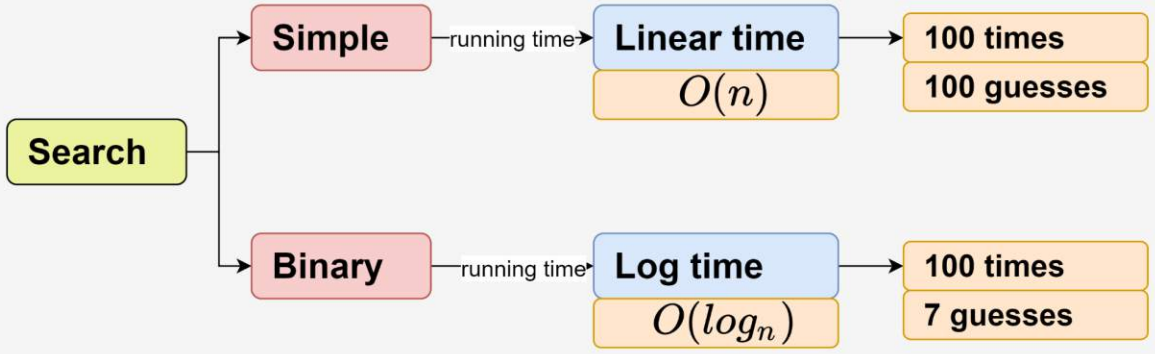
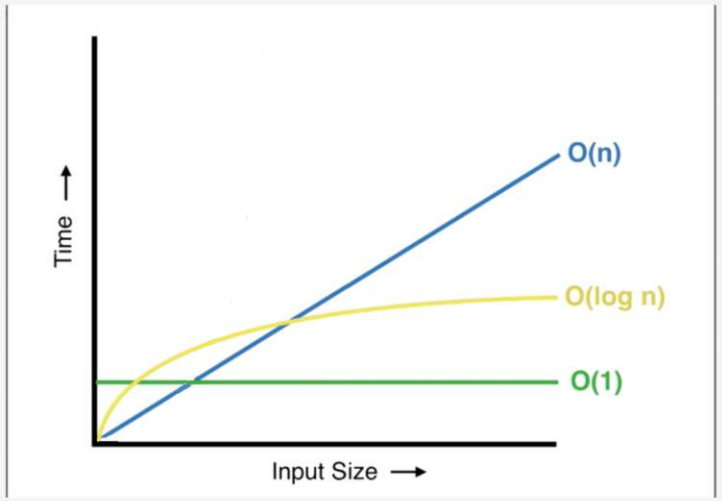


Figure 2.2



Big O notation

Is used to classify algorithms based on how fast they grow or decline, is used to analyze the efficiency of an algorithm as its input approaches infinity

Generally you want to use the most efficient algorithm.
back with the simple search we check each number, if this is a list of 100, it takes up to 100 guesses, if this is a list of 4 billion, then it takes 4 billion guesses, so the maximum number of guesses

Linear time

An algorithm is said to take linear time when the running time increases at most with the size of the input, this means, the maximum time is the same size as the size of the input.

Running time binary search

Binary search is different if the list is 100 items long, it takes at most 7 guesses, if the list is 4 billion items, it takes at most 32 guesses.

Logarithmic time of log time

An algorithm is said to take logarithmic time when the running time grows in to the logarithm of the input size.

2 Binary search By Aditya Y. Bhargava

Figure 2.1

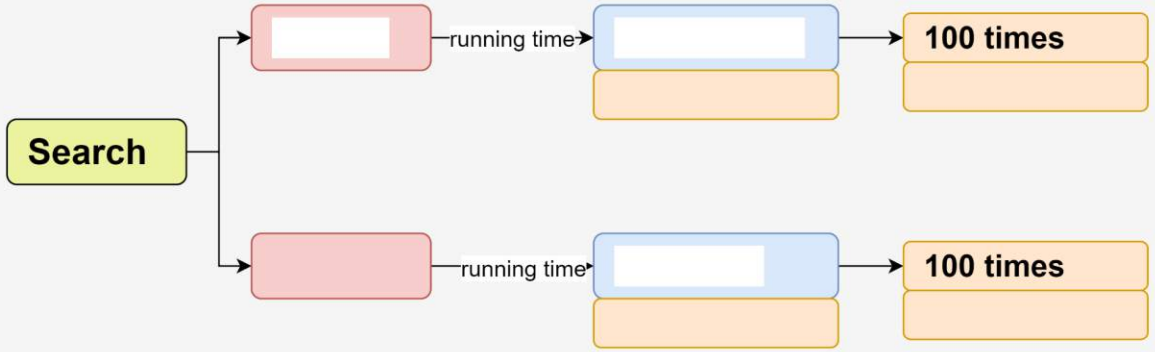
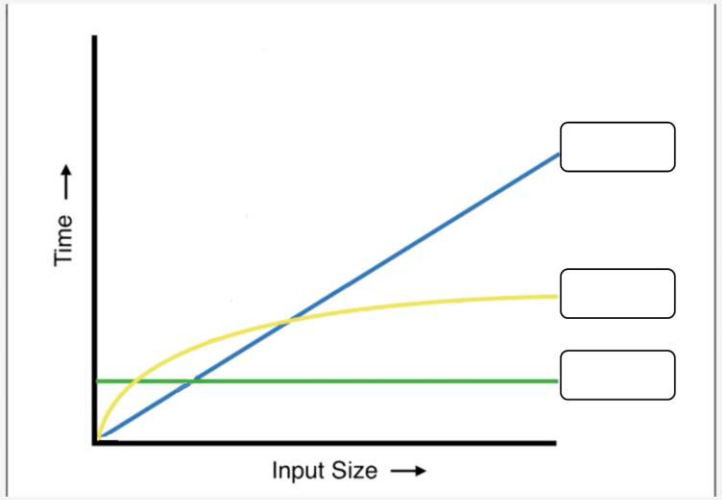


Figure 2.2



Big O notation

Is used to classify algorithms based on how fast they grow or decline, is used to analyze the efficiency of an algorithm as its input approaches

Algorithm running times grow at different rates

Consider binary and simple search, lets assume it takes 1 ms to check one element.

Figure 3.1

	SIMPLE SEARCH	BINARY SEARCH
100 ELEMENTS	100 ms	7 ms
10,000 ELEMENTS	10 seconds	14 ms
1,000,000,000 ELEMENTS	11 days	32 ms

As you can see as the number of times increase, binary search takes a little more time to run, but simple search takes a ***lot*** more time to run.

Importance of Big(O)

It is not enough to know how long an algorithm takes to run, you also need to know how the running times increases as the list size ***increases***.

Notation

Binary search need ***log n*** operations to check a list of size ***n***, the running time its then $O(\log_2 n)$.

Figure 3.2

Big O notation

$O(n)$

- it does → Tells you how **fast** the algorithm **grows**
- it does not → Tells you how much time the algorithm **last**
- it always look → Worst case scenario

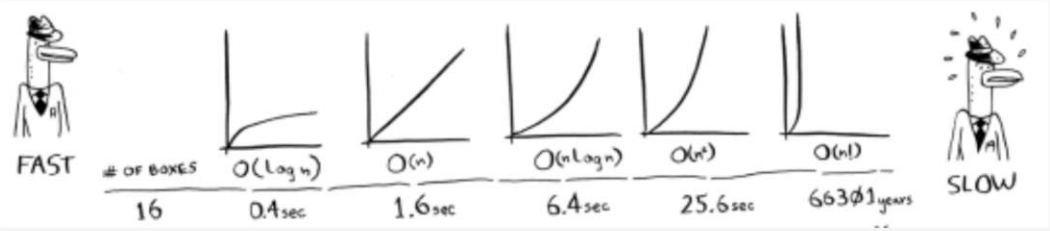
3 Binary search By Aditya Y. Bhargava

Common big O run times

Here are five big O run times you will encounter a lot, from fastest to slowest

$O(log \quad n)$	example →	**Binary search**
$O(n)$	example →	**Simple search**
$O(n * log \quad n)$	example →	**Quicksort**
$O(n^2)$	example →	**Selection sort**
$O(n!)$	example →	**Traveling salesperson**

How long will it take you if you are drawing a grid of 4x4, 16 boxes.



Give the run time for each of these scenarios in terms of Big O.

You have a name, and you want to find the person's phone numberin the phone book.	$O(log \quad n)$
You have a phone number, and you want to find the person's name in the phone book. (Hint: You'll have to search through the whole book!)	$O(n)$

Algorithm running times grow at different rates

Consider binary and simple search, lets assume it takes 1 ms to check one element.

Figure 3.1

	SIMPLE SEARCH	BINARY SEARCH
100 ELEMENTS	100 ms	7 ms
10,000 ELEMENTS	10 seconds	14 ms
1,000,000,000 ELEMENTS	11 days	32 ms

As you can see as the number of times increase, binary search takes a little more time to run, but simple search takes a more time to run.

Importance of Big(O)

It is not enough to know how long an algorithm takes to run, you also need to know how the running times increases as the list size .

Notation

Binary search need operations to check a list of size , the running time its then $O(\log_2 n)$.

Figure 3.2

Big O notation

$O(n)$

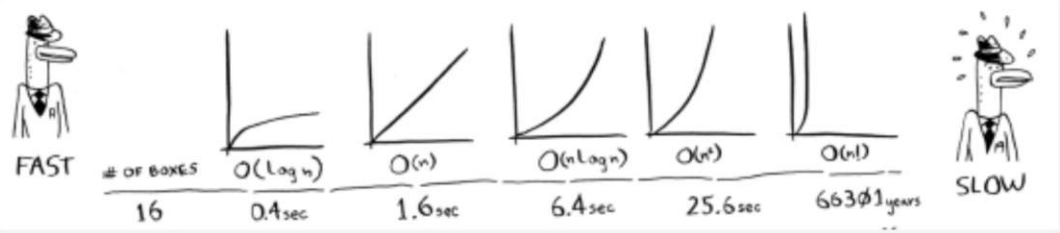
- it does → Tells you how the algorithm
- it does not → Tells you how much time the algorithm
- it always look → Worst case scenario

Common big O run times

Here are five big O run times you will encounter a lot, from fastest to slowest

$O(log \quad n)$	example →	
$O(n)$	example →	
	example →	**Quicksort**
	example →	**Selection sort**
	example →	**Traveling salesperson**

How long will it take you if you are drawing a grid of 4x4, 16 boxes.



Give the run time for each of these scenarios in terms of Big O.

You have a name, and you want to find the person's phone numberin the phone book.	
You have a phone number, and you want to find the person's name in the phone book. (Hint: You'll have to search through the whole book!)	

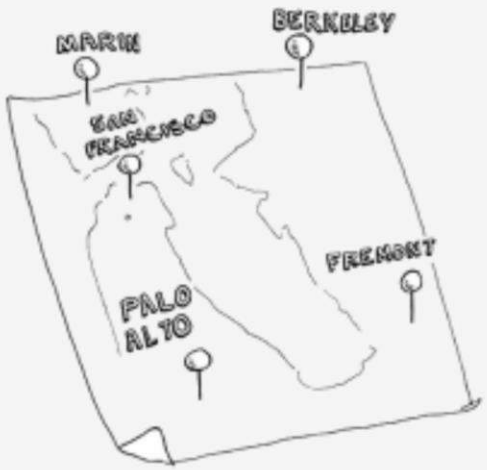
4 The traveling salesperson

By Aditya Y. Bhargava

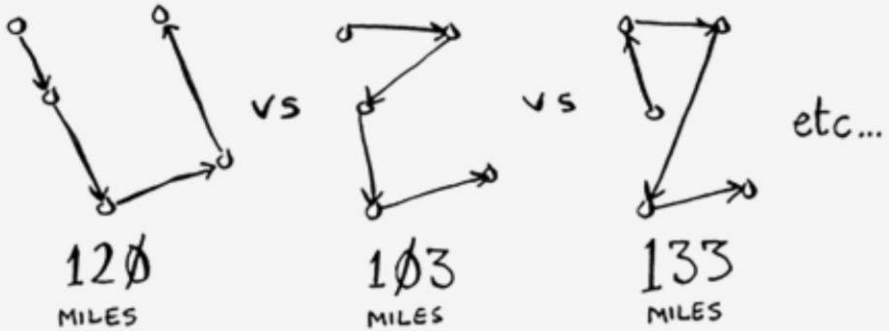
You may probably think *"there is no way i'll ever run into an algorithm that takes **O(n!)** time"*.

However

Here is an example of an algorithm with a really bad running time.

You have a sales person	The sales person has to go to five cities.
	

This salesperson wants to hit all five cities while traveling the minimum distance, that can be many options.



4 Binary search

By Aditya Y. Bhargava

He then picks the path with the lowest distance.
In 5 cities we have 120 permutations, so it will take 120 operations to solve this problem.

for 6 cities it will take 720 operations.

for 7 cities it will take 5,040 operations.

CITIES	OPERATIONS
6	720
7	5040
8	40320
...	...
15	1,307,674,368,000
...	...
30	265,252,859,812,191,058,636,308,480,000,000

Running time

This means it will take **n!**(n factorial operation to compute the results. This is **O(n!)** time, once you are dealing with 100 cities, its impossible to calculate the answer in time.

Summary

1	Binary search is a lot **faster** than simple search.
2	**O(Log n)** is faster than **O(n)**, but it gets a lot faster the more items you have.
3	Algorithm speed is not measured in **seconds**.
4	Algorithm times are measured in terms of **growth** of an algorithm.
5	Algorithm times are written in **big O** notation

The traveling salesperson

By Aditya Y. Bhargava

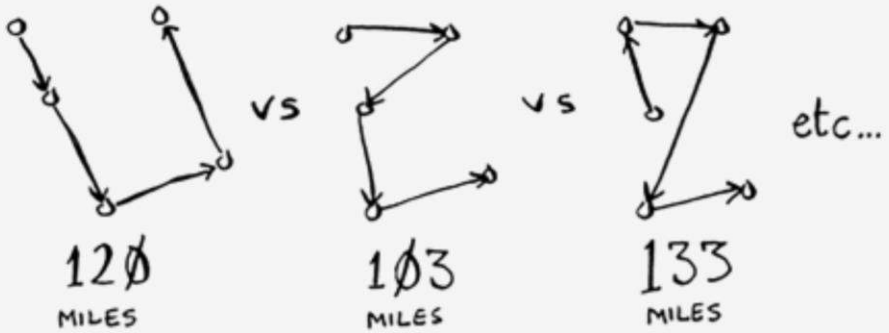
You may probably think *"there is no way i'll ever run into an algorithm that takes* ____ *time"*.

However

Here is an example of an algorithm with a really bad running time.

You have a sales person	The sales person has to go to five cities.

This salesperson wants to hit all five cities while traveling the minimum distance, that can be many options.



Binary search

By Aditya Y. Bhargava

He then picks the path with the lowest distance.
In 5 cities we have 120 permutations, so it will take 120 operations to solve this problem.

for 6 cities it will take 720 operations.

for 7 cities it will take 5,040 operations.

CITIES	OPERATIONS
6	720
7	5040
8	40320
...	...
15	1,307,674,368,000
...	...
30	265,252,859,812,191,058,636,308,480,000,000

Running time

This means it will take **n!**(n factorial operation to compute the results. This is **O(** **)** time, once you are dealing with 100 cities, its impossible to calculate the answer in time.

Summary

1	Binary search is a lot ____ than simple search.
2	**O(** ____ **)** is faster than **O(** **)**, but it gets a lot faster the more items you have.
3	Algorithm speed is not measured in ____.
4	Algorithm times are measured in terms of ____ of an algorithm.
5	Algorithm times are written in ____ notation

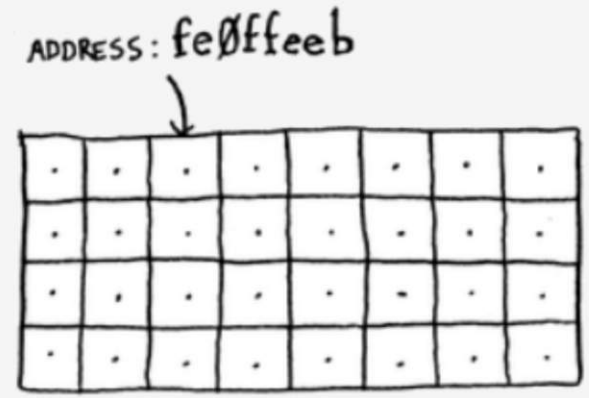
Image you have a drawer

Each drawer can hold one element, and you went to store two things, so you ask for two drawers.

You store your two things in her.

5 How memory works By Aditya Y. Bhargava

This is basically how your computer works, your computer looks like a giant set of drawers and each drawer has an address.



Address

fe0ffeeb is the address of a slot in memory.
Each time you want to store an item in memory, you ask the computer for some space/drawer, and it gives you an address where you can store it

Storing multiple items

There are two basics ways to store multiple items those are **arrays** and **lists**.

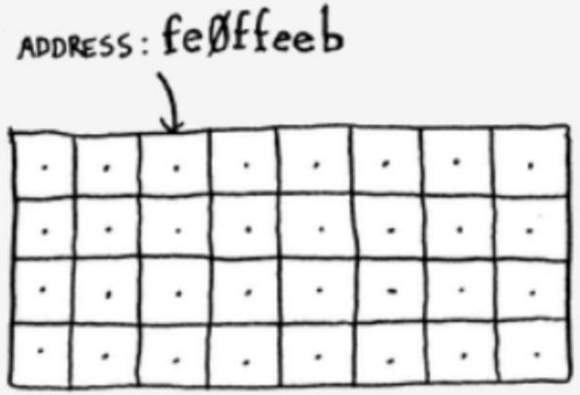
Image you have a drawer

Each drawer can hold one element, and you went to store two things, so you ask for two drawers.

You store your two things in her.

5 How memory works By Aditya Y. Bhargava

This is basically how your computer works, your computer looks like a giant set of drawers and each drawer has an address.



Address

fe0ffeeb is the address of a slot in memory.
Each time you want to store an item in memory, you ask the computer for some space/drawer, and it gives you an address where you can store it

Storing multiple items

There are two basics ways to store multiple items those are _____ and _____.

Sometimes you need to **store** a list of elements in memory.

Suppose you are writing an app to manage your to-dos.

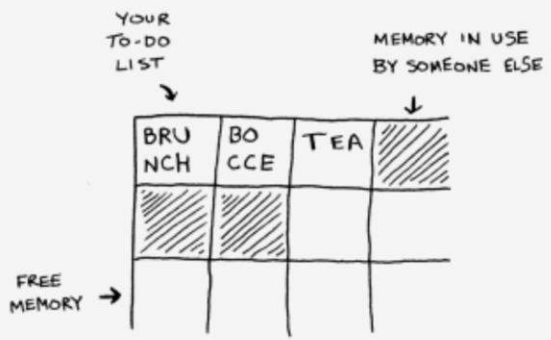
I guess you will want to store the to-dos as a list in memory.

¿What should we use array or linked list?

Array

Array is **easier** to grasp, using arrays means all your task are stored **continuously** in memory, that is right next to each other.

Now suppose you want to add a fourth task.
well you cant because the next drawer is taken up by someone else!.



Its like going to a movie with your friends but there is no place for them to sit together!

You will need a different method in that case.

Because adding items to an array can be a pain, and if you really want a new spot of memory every time you want to add another value, this move will be really slow.

Well what about...

Hold seats, that is telling the computer to reserve for example 10 slots.

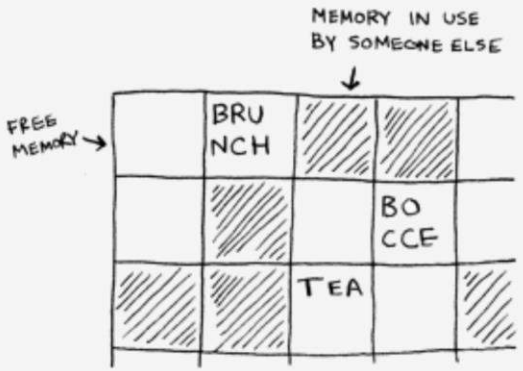
However what can happen is that.

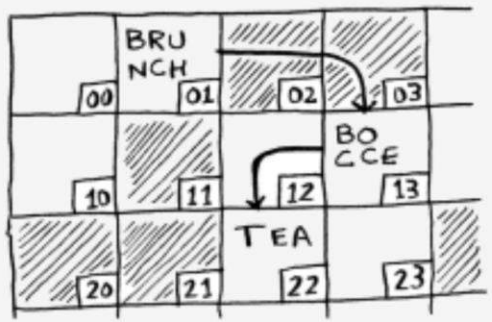
A	You may not **need** the extra slots.
B	You may **add** more items and have to move anyway.

Linked lists

With linked list your items can be anywhere in memory

each item, stores the **address** of the next item in the lists.





Its like a treasure hunt, you go to the first address 132, and then it says the next is in address 847.

Adding an item to a linked list is easy. You add it in the memory and store the **address** with the previous item.

Sometimes you need to _____ a list of elements in memory.

Suppose you are writing an app to manage your to-dos.

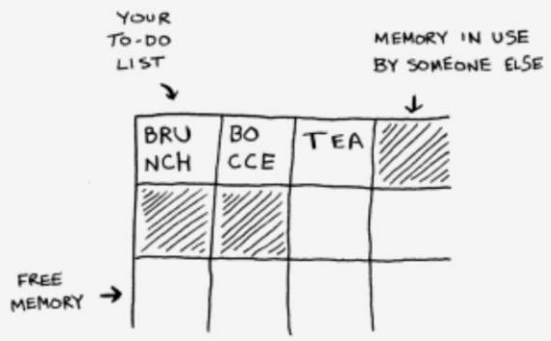
I guess you will want to store the to-dos as a list in memory.

¿What should we use array or linked list?

Array

Array is _____ to grasp, using arrays means all your task are stored _____ in memory, that is right next to each other.

Now suppose you want to add a fourth task.
well you cant because the next drawer is taken up by someone else!.



Its like going to a movie with your friends but there is no place for them to sit together!

You will need a different method in that case.

Because adding items to an array can be a pain, and if you really want a new spot of memory every time you want to add another value, this move will be really slow.

Well what about...

Hold seats, that is telling the computer to reserve for example 10 slots.

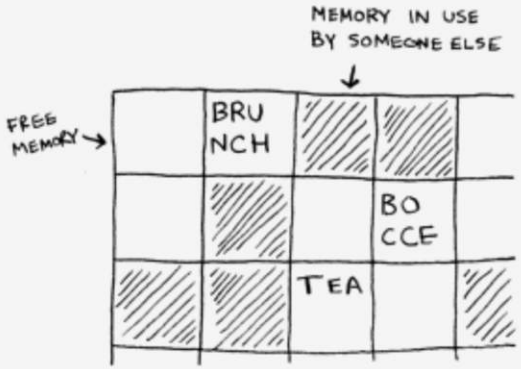
However what can happen is that.

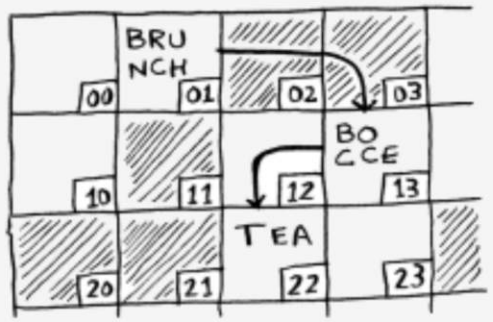
A	You may not _____ the extra slots.
B	You may _____ more items and have to move anyway.

Linked lists

With linked list your items can be anywhere in memory

each item, stores the _____ of the next item in the lists.





Its like a treasure hunt, you go to the first address 132, and then it says the next is in address 847.

Adding an item to a linked list is easy. You add it in the memory and store the _____ with the previous item.

Consider a page which has a top 10 best villians

Instead of putting the entire list on a page, they only show, one item on each page and make you click "**next**" to get the next item in the list.

This technique gives this websites 10 whole pages on which to show you ads.

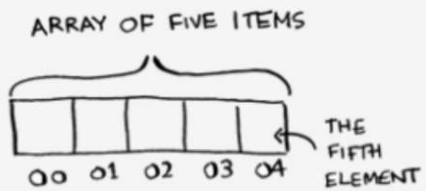
But this is boring...

Linked lists have a similar problem, suppose you want to read the last item in a linked list, you just cant, you have to go from item #1. To get the **address** for item 2, and keep going this until the last item.

Arrays are different

You **know** the address for every item in your array.

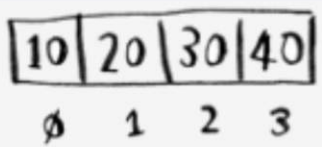
For example what is the address of item #5, simple math tell us; its **04**.



Terminology

The elements in an array are numbered, this numbering starts at 0, not 1.

In this array, 20 is at position **1** and 10 is at position **0**.



This usually throws new programmers off.

However

Starting at 0 makes all kinds of array based code easier to write.

Index

The position of an element its called its **index**, so instead of saying "20 is at position 1", the correct way is "20 is at **index** 1".
Index will mean position.

Common run times for common operations on arrays and lists.

	ARRAYS	LISTS
READING	O(1)	O(n)
INSERTION	O(n)	O(1)

O(n) = LINEAR TIME
O(1) = CONSTANT TIME

Exercise

Suppose you are building an app to keep track of your finances

1. GROCERIES
2. MOVIE
3. SFBC MEMBERSHIP

Everyday you write down everything you spend money on, so, you have a lots of inserts and a few reads.

¿Should you use an array or a list?

Consider a page which has a top 10 best villians

Instead of putting the entire list on a page, they only show, one item on each page and make you click " " to get the next item in the list.

This technique gives this websites 10 whole pages on which to show you ads.

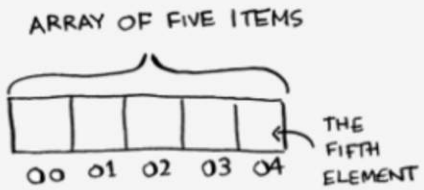
But this is boring...

Linked lists have a similar problem, suppose you want to read the last item in a linked list, you just cant, you have to go from item #1. To get the for item 2, and keep going this until the last item.

Arrays are different

You **know** the address for every item in your array.

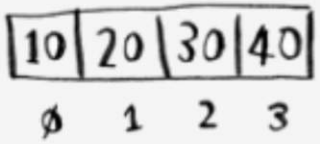
For example what is the address of item #5, simple math tell us; its .



Terminology

The elements in an array are numbered, this numbering starts at 0, not 1.

In this array, 20 is at position and 10 is at position .



This usually throws new programmers off.

However

Starting at 0 makes all kinds of array based code easier to write.

Index

The position of an element its called its , so instead of saying "20 is at position 1", the correct way is "20 is at 1".
Index will mean position.

Common run times for common operations on arrays and lists.

	ARRAYS	LISTS
READING	O(1)	O(n)
INSERTION	O(n)	O(1)

O(n) = LINEAR TIME
O(1) = CONSTANT TIME

Exercise

Suppose you are building an app to keep track of your finances

1. GROCERIES
2. MOVIE
3. SFBC MEMBERSHIP

Everyday you write down everything you spend money on, so, you have a lots of inserts and a few reads.

¿Should you use an array or a list?

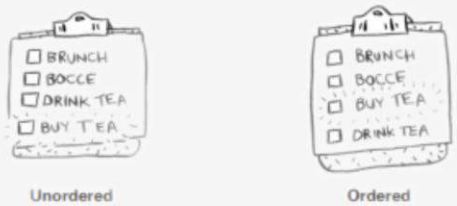
8 Insert into the middle of a list

By Aditya Y. Bhargava

Suppose you want your to-do list to work more like a calendar.

Earlier we were adding things to the end of the list.

But now you choose the order in which they should be done.



What is better for this

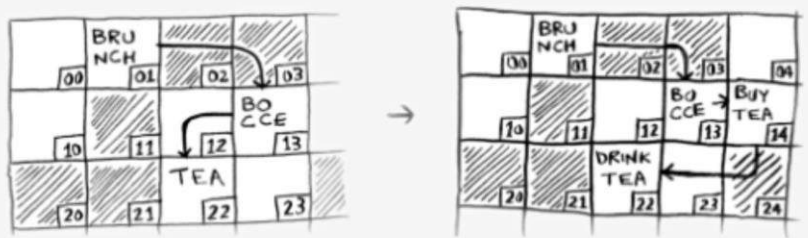
If you want to insert an element in the middle: ¿arrays or list?

Arrays

What happens in arrays is that you have to shift all the rest of the elements **down**

Lists

With lists, its easy as changing what the previous element **points** to, list are better if you want to insert elements into the **middle**.



8 Deletions

By Aditya Y. Bhargava

If you want to delete an element, lists are better, that is because you just need to change the previous **pointer**.

However

With array you will need to move everything **up** when you delete an element.

Common run times on common operations

deletions will always **work**, insertions can **fail** sometimes when there is no space left in memory, but you can always delete an element.

Its worth mentioning that insertions and deletions are **O(1)** time only if you can instantly access the element to be deleted.

	ARRAYS	LISTS
READING	O(1)	O(n)
INSERTION	O(n)	O(1)
DELETION	O(n)	O(1)

It is common

To keep track of the first and last item in a linked list.

What should i use

It depends on the use case, Arrays see a lot of use because they allow random acess, there are two types of access.

Random access	Sequential access
Can jump directly to the 10th element	Reading elements 1 by 1 like linked lists

8 Insert into the middle of a list

By Aditya Y. Bhargava

Suppose you want your to-do list to work more like a calendar.

Earlier we were adding things to the end of the list.

But now you choose the order in which they should be done.

What is better for this

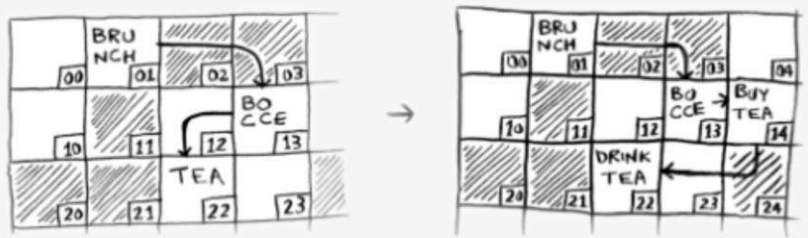
If you want to insert an element in the middle: ¿arrays or list?

Arrays

What happens in arrays is that you have to shift all the rest of the elements _____.

Lists

With lists, its easy as changing what the previous element _____ to, list are better if you want to insert elements into the _____.



8 Deletions

By Aditya Y. Bhargava

If you want to delete an element, lists are better, that is because you just need to change the previous _____.

However

With array you will need to move everything _____ when you delete an element.

Common run times on common operations

deletions will always _____, insertions can _____ sometimes when there is no space left in memory, but you can always delete an element.

Its worth mentioning that insertions and deletions are _____ time only if you can instantly access the element to be deleted.

	ARRAYS	LISTS
READING	O(1)	O(n)
INSERTION	O(n)	O(1)
DELETION	O(n)	O(1)

It is common

To keep track of the first and last item in a linked list.

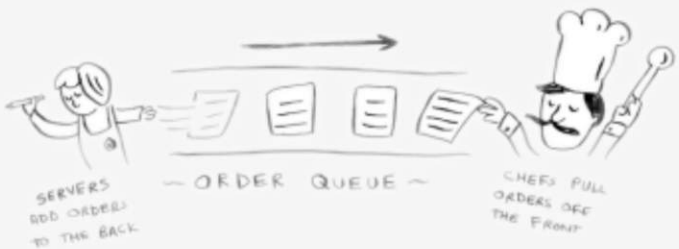
What should i use

It depends on the use case, Arrays see a lot of use because they allow random acess, there are two types of access.

_____	_____
Can jump directly to the 10th element	Reading elements 1 by 1 like linked lists

2.2

Suppose you are building an app for restaurants to take customer orders. your app needs to store a list of orders. Servers keep adding orders to this list, and chef take orders off the list and make them.



¿Would you use an array or linked list to implement this queue?

I would use a linked list, because there is a lot of inserts happening and because the chef always takes the first order off the queue you don't need to search or random access(Arrays).

2.3

Let's run a thought experiment. Suppose Facebook keeps a list of usernames. When someone tries to log in to Facebook, a search is done for their username. If their name is in the list of usernames, they can log in. People log in to Facebook pretty often, so there are a lot of searches through this list of usernames. Suppose Facebook uses binary search to search the list. Binary search needs random access—you need to be able to get to the middle of the list of usernames instantly.

¿Would you use an array or linked list to implement this system?

I would use a sorted array, arrays give you random access this way you can get the middle of the array instantly.

9 Excercises By Aditya Y. Bhargava

2.4

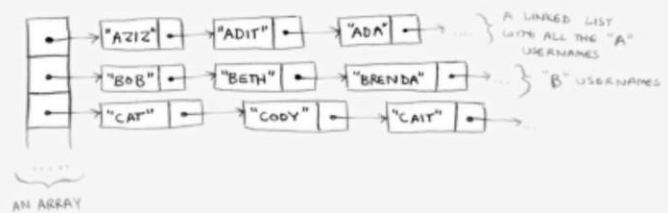
People sign up for Facebook pretty often, too. Suppose you decided to use an array to store the list of users. What are the downsides of an array for inserts? In particular, suppose you're using binary search to search for logins

¿What happens when you add new users to an array?

It will be slow, because the inserts in array.
Also since we use binary search we will need to sort the array every time a name is inserted.

2.5

In reality, Facebook uses neither an array nor a linked list to store user information. Let's consider a hybrid data structure: an array of linked lists. You have an array with 26 slots. Each slot points to a linked list. For example, the first slot in the array points to a linked list containing all the usernames starting with a. The second slot points to a linked list containing all the usernames starting with b, and so on.

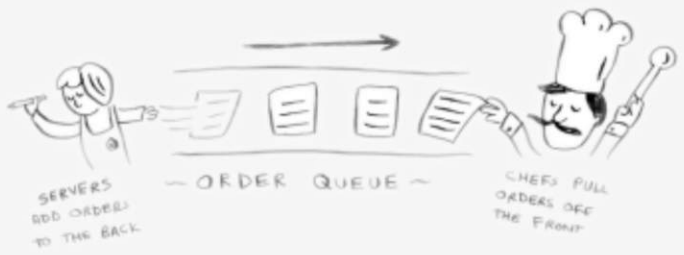


¿Is it slower or faster than each for searching and inserting?

In the matter of searching it will be slower than arrays, but faster than pure linked lists in the matter of insert, it will be faster than an array and the same amount of time as linked lists.

2.2

Suppose you are building an app for restaurants to take customer orders. your app needs to store a list of orders. Servers keep adding orders to this list, and chef take orders off the list and make them.



¿Would you use an array or linked list to implement this queue?

I would use a _____, because there is a lot of inserts happening and because the chef always takes the first order off the queue you don't need to _____ or _____ (_____).

2.3

Let's run a thought experiment. Suppose Facebook keeps a list of usernames. When someone tries to log in to Facebook, a search is done for their username. If their name is in the list of usernames, they can log in. People log in to Facebook pretty often, so there are a lot of searches through this list of usernames. Suppose Facebook uses binary search to search the list. Binary search needs random access—you need to be able to get to the middle of the list of usernames instantly.

¿Would you use an array or linked list to implement this system?

I would use a _____, _____ give you _____ this way you can get the middle of the array instantly.

2.4

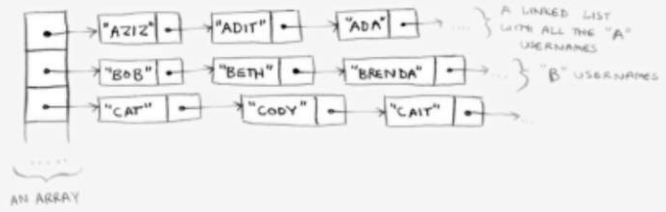
People sign up for Facebook pretty often, too. Suppose you decided to use an array to store the list of users. What are the downsides of an array for inserts? In particular, suppose you're using binary search to search for logins

¿What happens when you add new users to an array?

It will be slow, because the inserts in _____.
Also since we use binary search we will need to sort the array every time a name is inserted!

2.5

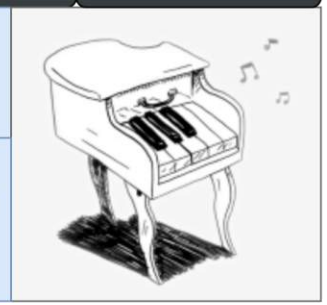
In reality, Facebook uses neither an array nor a linked list to store user information. Let's consider a hybrid data structure: an array of linked lists. You have an array with 26 slots. Each slot points to a linked list. For example, the first slot in the array points to a linked list containing all the usernames starting with a. The second slot points to a linked list containing all the usernames starting with b, and so on.



¿Is it slower or faster than each for searching and inserting?

In the matter of _____ it will be slower than arrays, but faster than pure linked lists in the matter of _____, it will be faster than an array and the same amount of time as linked lists.

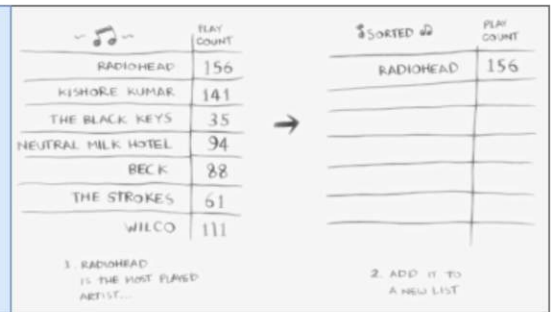
Suppose you have a bunch of music on your computer.
For each artist, you have a play count.



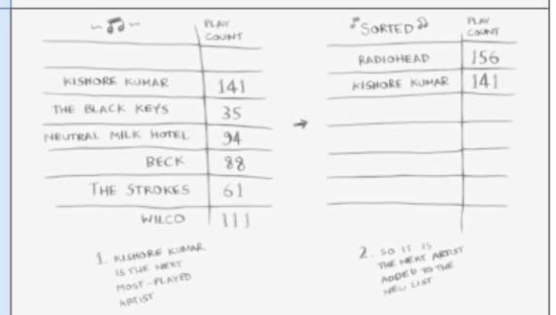
You want to sort this list from most to least played, so that you can rank your favorite artists.

¿How can you do it?

One way is to go through the list and find the most-played artist. Add that artist to a new list.

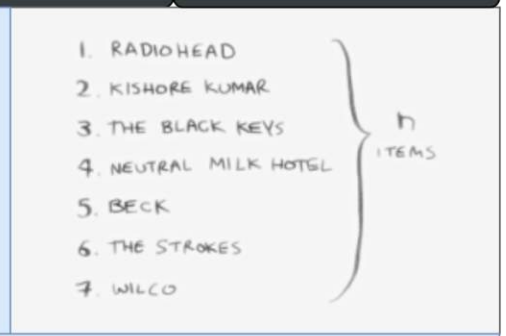


Do it again to find the next-most-played artist.

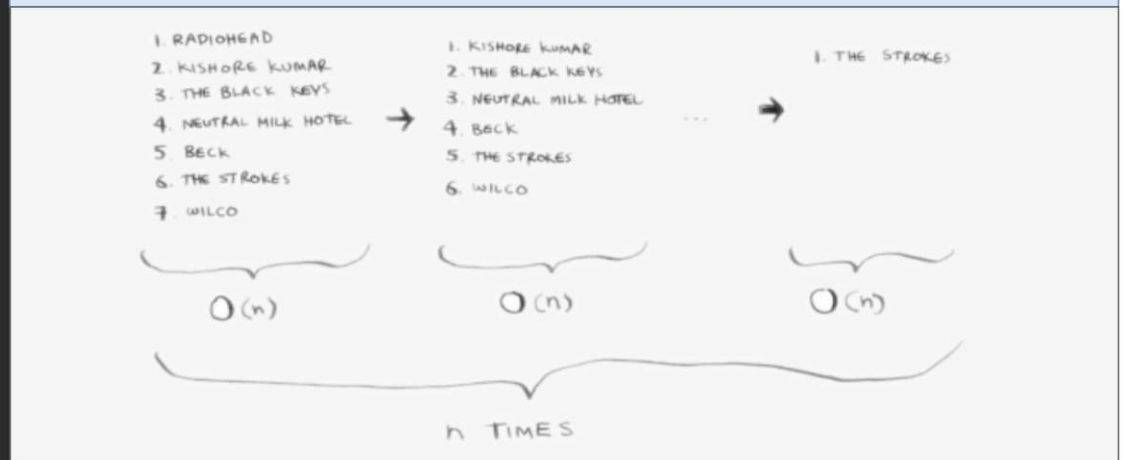


Keep doing this, and you'll end up with a sorted list.

Let's put on our computer science hats and see how long this will take to run. Remember that O(n) time means you touch every element in a list once. For example, running simple search over the list of artists means looking at each artist once.



To find the artist with the highest play count, you have to check each item in the list. This takes O(n) time, as you just saw. So you have an operation that takes O(n) time, and you have to do that n times:



his takes O(n × n) time or $O(n^2)$ time.

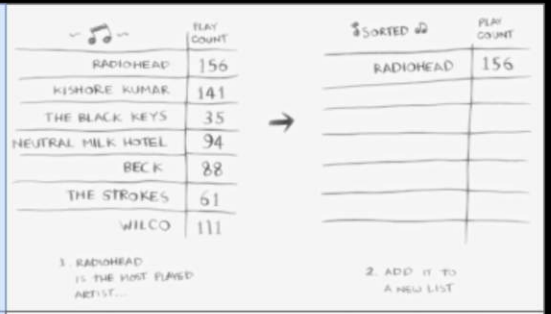
Selection sort is a neat algorithm, but it's not very fast. Quicksort is a faster sorting algorithm that only takes O(n log n) time.

Suppose you have a bunch of music on your computer.
For each artist, you have a play count.

You want to sort this list from most to least played, so that you can rank your favorite artists.

¿How can you do it?

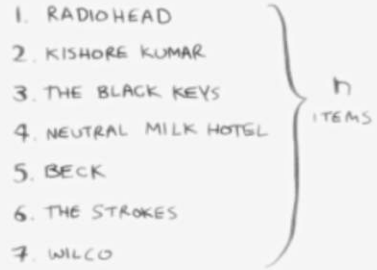
One way is to go through the list and find the most-played artist. Add that artist to a new list.



Do it again to find the next-most-played artist.

Keep doing this, and you'll end up with a sorted list.

Let's put on our computer science hats and see how long this will take to run. Remember that O(n) time means you touch every element in a list . For example, running simple search over the list of artists means looking at each artist once.



To find the artist with the highest play count, you have to check each item in the list. This takes time, as you just saw. So you have an operation that takes time, and you have to do that n times:

his takes time or time.

Selection sort is a neat algorithm, but it's not very fast. Quicksort is a faster sorting algorithm that only takes O(n log n) time.

Selection sort

By Aditya Y. Bhargava

Algorithm

```
function SelectionSort(Array)
    NewArray = [];
    for each i in Array
        Smallest = FindSmallest(Array);
        NewArray.Add(Smallest);
        Array.pop(Smallest);
    return NewArray;
```

Sorts an array

Find the index of the smallest array, and adds it to the new array.

remove element from array

Now we define the function FindSmallest() array.

Algorithm 2

```
function FindSmallest(Array)
    Smallest = Array[0];
    Smallest_index = 0;
    for each i in Array
        if (Array[i] < Smallest)
            Smallest =Array[i];
            Smallest_index = i;
    return NewArray;
```

This will store the smallest value and smallest infex of value

Check each element to see which is smallest.

Recap

By Aditya Y. Bhargava

1	Your computer memory is like a giant set of drawers
2	When you want to store multiple elements, use an array or a list.
3	With an array, all your elements are stored right next to each other.
4	With a list, elements are strewn all over, and one element stores the address of the next one.
5	Arrays allow fast reads
6	Linked list allow fast inserts and deletes
7	All elements in the array should be the same type(all ints, all doubles, and so on)

Selection sort

By Aditya Y. Bhargava

Algorithm

```
function SelectionSort(Array)
    NewArray = [];
    for each i in Array
        Smallest = FindSmallest(Array);
        NewArray.Add(Smallest);
        Array.pop(Smallest);
    return NewArray;
```

Now we define the function FindSmallest() array.

Algorithm 2

```
function FindSmallest(Array)
    Smallest = Array[0];
    Smallest_index = 0;
    for each i in Array
        if (Array[i] < Smallest)
            Smallest =Array[i];
            Smallest_index = i;
    return NewArray;
```


Recap

By Aditya Y. Bhargava

1	Your computer memory is like a giant set of _____.
2	When you want to store multiple elements, use an _____ or a _____.
3	With an array, all your elements are stored right next to _____ _____.
4	With a list, elements are strewn all over, and one element stores the address of the next one.
5	Arrays allow fast _____
6	Linked list allow fast _____ and _____
7	All elements in the array should be the same _____ (all ints, all doubles, and so on)

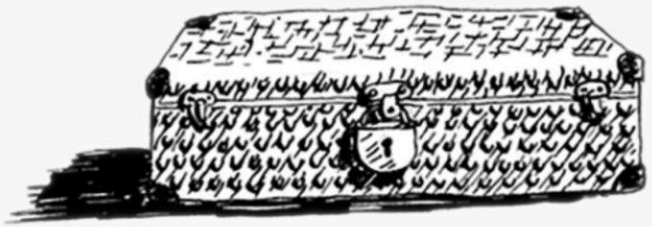
Es una manera elegante de resolver problemas, este suele ser divisivo en su apreciación, puedes amarlo o puedes odiarlo, esta sección incluye pseudocodigo.

Pseudocodigo

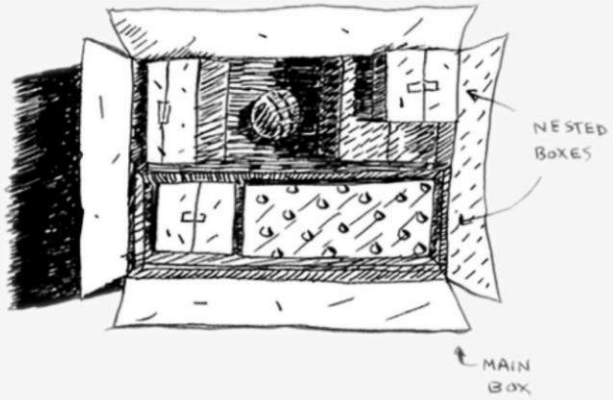
Es una descripción de alto nivel del problema que estas tratando de resolver en código, es escrito en forma de código pero esta hecho para parecer un discurso humano.

Explicación

Imagina que estas indagando en el ático de tu abuela, y te encuentras con una misteriosa maleta con un candado.

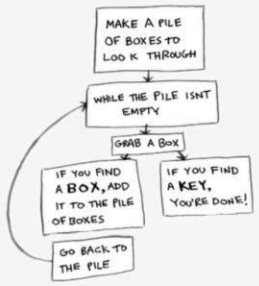


Tu abuela te dice que la llave para esa maleta probablemente se encuentre en esta caja:



Esta caja contiene más cajas, y esas cajas contienen aun más cajas y la llave se encuentra en una de ellas. ¿Qué algoritmos usarias?

Pensando en un efoque



¿Qué enfoque se te hizo más sencillo?

El primer enfoque utiliza un ciclo while, mientras que la pila no este vacia, toma una caja y busca a través de ella.

```
function Buscar(Box)
    Pila.add();
    while Pila no vacia
        Box = pila.pop();
        for each item in Box
            if (Item is a Box)
                Pile.add();
            else return Key;
```

La segunda usa recursión, recursión es cuando una función se llama sí mismo.

```
function Buscar(Box)
    for each item in Box
        if (Item is a Box)
            Buscar(Box);
        else return Key;
```

Ambos enfoques logran lo mismo, pero el segundo parece más claro.

¿Cuando usar recursión?

Recursion es usada cuando hace al algoritmo más claro, no hay beneficio en rendimiento al usar recursión, de hecho los ciclos algunas veces suelen ser mejor.

Leigh Caldwell en stackoverflow dice lo siguiente:

Los ciclos pueden lograr rendimiento a tu programa, pero la recursión puede lograr rendimiento a tu programador.

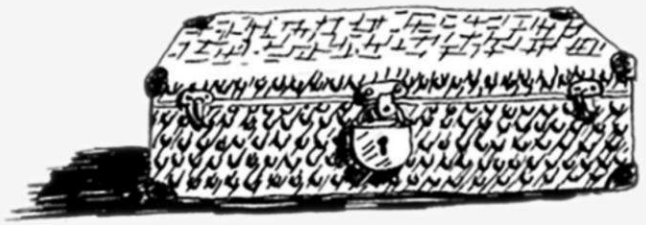
Es una manera elegante de resolver problemas, este suele ser divisivo en su apreciación, puedes amarlo o puedes odiarlo, esta sección incluye pseudocodigo.

Pseudocodigo

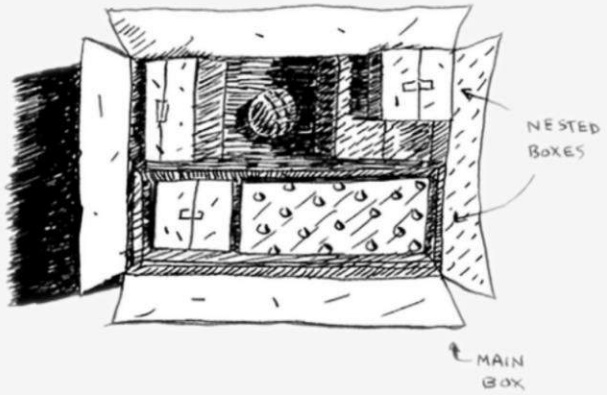
Es una descripción de alto nivel del problema que estas tratando de resolver en código, es escrito en forma de código pero esta hecho para parecer un [redacted].

Explicación

Imagina que estas indagando en el ático de tu abuela, y te encuentras con una misteriosa maleta con un candado.



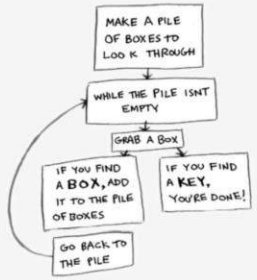
Tu abuela te dice que la llave para esa maleta probablemente se encuentre en esta caja:

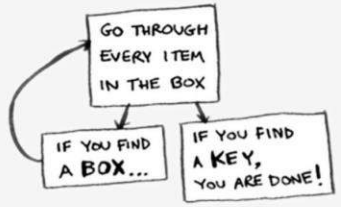


Esta caja contiene más cajas, y esas cajas contienen aun más cajas y la llave se encuentra en una de ellas. ¿Qué algoritmos usarias?

Pensando en un efoque

3 Recursión By Aditya Y. Bhargava





¿Qué enfoque se te hizo más sencillo?

El primer enfoque utiliza un ciclo while, mientras que la pila no este vacia, toma una caja y busca a través de ella.

```
function Buscar(Box)
    Pila.add();
    while Pila no vacia
        Box = pila.pop();
        for each item in Box
            if (Item is a Box)
                Pile.add();
            else return Key;
```

La segunda usa recursión, recursión es cuando una función se llama sí mismo.

```
function Buscar(Box)
    for each item in Box
        if (Item is a Box)
            Buscar(Box);
        else return Key;
```

Ambos enfoques logran lo mismo, pero el segundo parece más claro.

¿Cuando usar recursión?

Recursion es usada cuando hace al algoritmo más [redacted], no hay beneficio en rendimiento al usar recursión, de hecho los ciclos algunas veces suelen ser mejor.

Leigh Caldwell en stackoverflow dice lo siguiente:

Los ciclos pueden lograr rendimiento a tu programa, pero la recursión puede lograr rendimiento a tu programador.

3.1 Caso base y caso recursivo

By Aditya Y. Bhargava

Ya que una función recursiva se llama a si misma, existe el riesgo de escribir una función con un loop infinito.

Supon que escribes una función que imprime un contador:

```
> 3 ... 2 ... 1
```

Puede escribirse así:

```
function Contador(i)
    Imprimir i;
    Contador(i - 1)
```

Infinite loop

Tu te daras cuenta del problema, esta función correra siempre!

Cuando escribes una función recursiva, tienes que decirle cuando parar de recursar.
Es por esto que toda función recursiva tiene 2 partes: el caso base y el caso recursivo.

El caso recursivo

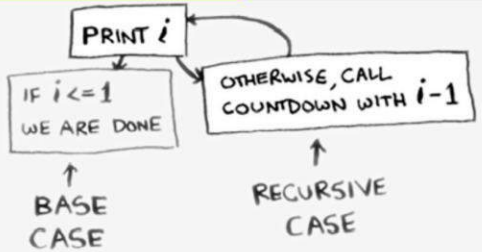
Es cuando la función se llama a si mismo.

El caso base

Es cuando la función no se llama de nuevo, de esta manera no entra en el loop infinito.

Añadamos el caso base a nuestra función

```
function Contador(i)
    Imprimir i;
        if (i ≤ 0)
            return;
        else Contador(i - 1);
```



3.2 The Stack

By Aditya Y. Bhargava

Call stack es un concepto importante en programación en general y sobre todo al usar recursión.

Imagina que estas en una carne asada y tu tienes una lista de pendientes en una pila de notas amarillas.

En esta **Pila**, cuando insertas una nota lo haces a la parte superior, cuando quieres leer una nota solo lees la que se encuentra en la parte superior y es removida de la lista.

Push y Pop

Push (insertar)	Pop (Remover y leer)
PUSH (ADD A NEW ITEM TO THE TOP)	POP (REMOVE THE TOPMOST ITEM AND READ IT)

Veamoslo en acción

3.1 Caso base y caso recursivo

By Aditya Y. Bhargava

Ya que una función recursiva se llama a si misma, existe el riesgo de escribir una función con un loop infinito.

Supon que escribes una función que imprime un contador:

```
> 3 ... 2 ... 1
```

Puede escribirse así:

```
function Contador(i)
    Imprimir i;
    Contador(i - 1)
```

Infinite loop

Tu te daras cuenta del problema, esta función correra !

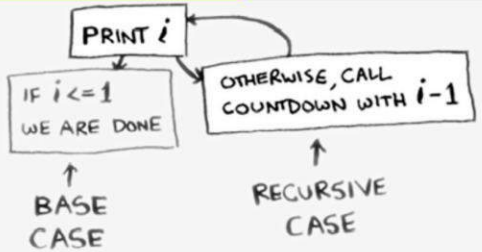
Cuando escribes una función recursiva, tienes que decirle cuando parar de recursar.
Es por esto que toda función recursiva tiene 2 partes: el caso base y el caso recursivo.

Es cuando la función se llama a si mismo.

Es cuando la función no se llama de nuevo, de esta manera no entra en el loop infinito.

Añadamos el caso base a nuestra función

```
function Contador(i)
    Imprimir i;
        if (i ≤ 0)
            return;
        else Contador(i - 1);
```



3.2 The Stack

By Aditya Y. Bhargava

Call stack es un concepto importante en programación en general y sobre todo al usar recursión.

Imagina que estas en una carne asada y tu tienes una lista de pendientes en una pila de notas amarillas.

En esta , cuando insertas una nota lo haces a la parte superior, cuando quieres leer una nota solo lees la que se encuentra en la parte superior y es removida de la lista.

Push y Pop

Push ( )	Pop (y)
PUSH (ADD A NEW ITEM TO THE TOP)	POP (REMOVE THE TOPMOST ITEM AND READ IT)

Veamoslo en acción

3.3 Call Stack

By Aditya Y. Bhargava

La computadora usa un plan interno llamado **Call stack**, para entender su funcionamiento, analizaremos una función:

```
function Saludar(Nombre)
    Imprimir "Hola" + Nombre;
    Greet2(Nombre);
    Imprimir "Hola" + Nombre;
    Bye(Nombre);
```

Esta función como podemos ver llama a otras 2 funciones, veamos que pasa cuando llamamos a una función.

1. Supon que llamas "Greet("Ale")". primero tu computadora reservara un espacio de memoria para esa función.

2. Ahora usaremos la memoria, el nombre de la valor de la variable es "Ale".

3. Cada vez que llamas a una función tu computadora guarda los valores de todas las variables en memoria, despues llamas a Greet2("Ale").

4. Tu computador usa una pila para estas cajas, la segunda caja fue añadida arriba de la pila, cuando esta termina, es removida de la pila.

Esta nos da una idea funtamental

Cuando llamas a una función desde otra función, la función que llama es **pausada** en un estado de completación parcial y todas las variables de la primera función siguen grabadas en memoria.

3.3 Call Stack

By Aditya Y. Bhargava

5. Ahora se llama a la funcion bye()

6. Y cuando se remueve la pila

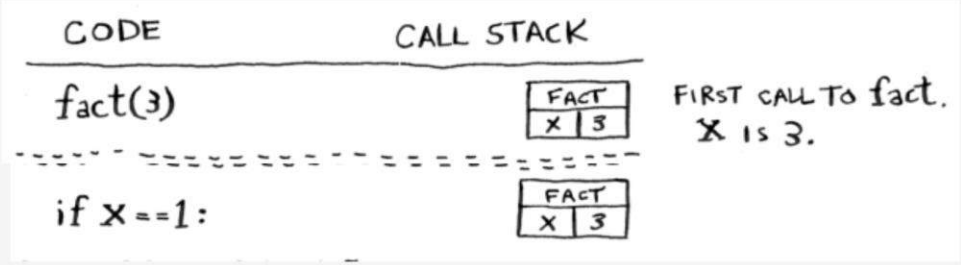
Esto es el Call stack y es usado para guardar las variables de las múltiples funciones que hemos llamado.

Call stack con recursión

Las funciones recursivas también utilizan el call stack, veamos la siguiente función factorial.

```
function factorial(X)
    if(x == 1);
        return 1;
    else:
     return X * factorial(X-1);
```

Imaginemos que llamamos la función factorial(3), veamos esta llamada linea por linea, recordemos que la caja de hasta arriba es la caja en la que estamos en el presente



La computadora usa un plan interno llamado _____, para entender su funcionamiento, analizaremos una función:

```
function Saludar(Nombre)
    Imprimir "Hola" + Nombre;
    Greet2(Nombre);
    Imprimir "Hola" + Nombre;
    Bye(Nombre);
```

Esta función como podemos ver llama a otras 2 funciones, veamos que pasa cuando llamamos a una función.

1 Supon que llamas "Greet("Ale")". primero tu computadora reservara un espacio de memoria para esa función.

2 Ahora usaremos la memoria, el nombre de la valor de la variable es "Ale".

3 Cada vez que llamas a una función tu computadora guarda los valores de todas las variables en memoria, despues llamas a Greet2("Ale").

4 Tu computador usa una pila para estas cajas, la segunda caja fue añadida arriba de la pila, cuando esta termina, es removida de la pila.

Esta nos da una idea funtamental

Cuando llamas a una función desde otra función, la función que llama es _____ en un estado de completación parcial y todas las variables de la primera función siguen grabadas en memoria.

5 Ahora se llama a la funcion bye()

6 Y cuando se remueve la pila

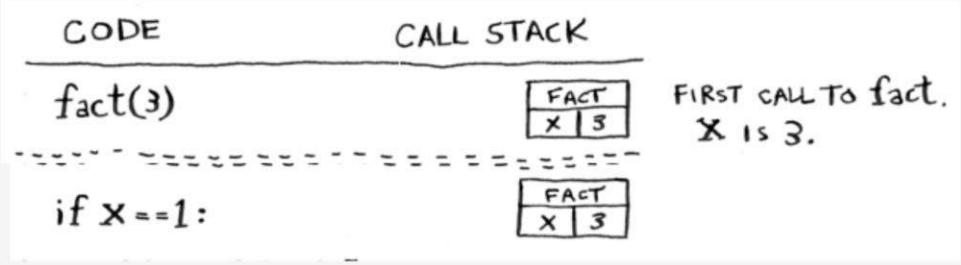
Esto es el Call stack y es usado para guardar las variables de las múltiples funciones que hemos llamado.

Call stack con recursión

Las funciones recursivas también utilizan el call stack, veamos la siguiente función factorial.

```
function factorial(X)
    if(x == 1);
        return 1;
    else:
     return X * factorial(X-1);
```

Imaginemos que llamamos la función factorial(3), veamos esta llamada linea por linea, recordemos que la caja de hasta arriba es la caja en la que estamos en el presente



3.3 Call Stack

By Aditya Y. Bhargava

Codigo	Stack	Comentario
fact(3)	fact x 3	Primera llamada a la función factorial, x es 3.
if x == 1	fact x 3	
else	fact x 3	
return x * fact(x-1)	fact x 2 fact x 3	una llamada recursiva!
if x == 1	fact x 2 fact x 3	Ahora estamos en la segunda llamada de la llamada al factorial, X es 2.
else	fact x 2 fact x 3	Ambas llamadas a las funciones tienen una variable llamada x, y es diferente en ambos.

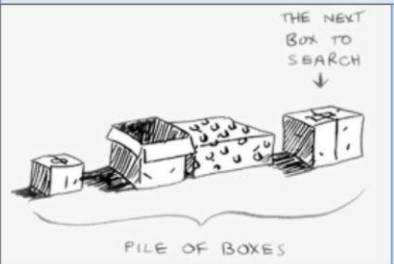
3.3 Call Stack

By Aditya Y. Bhargava

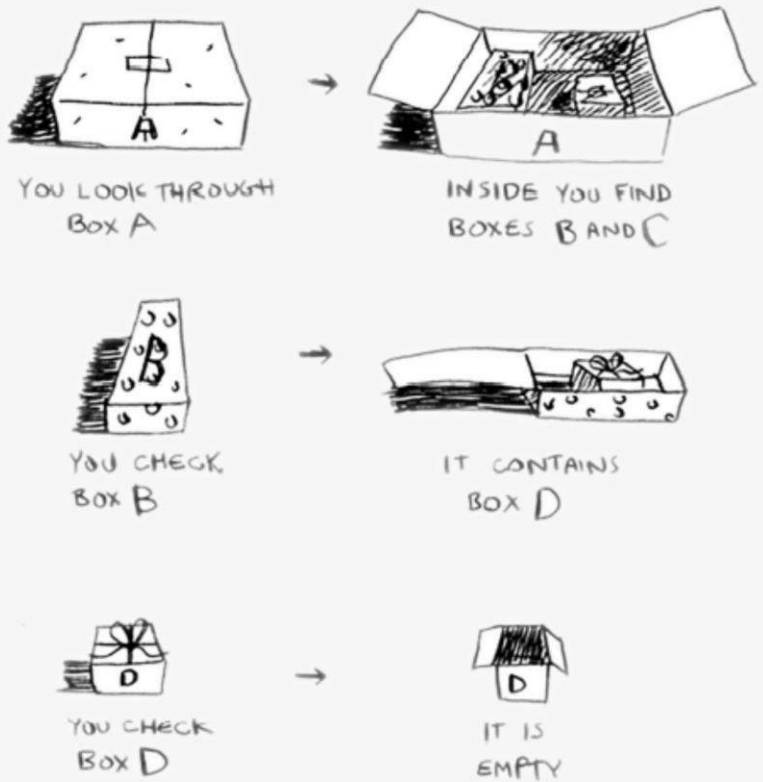
Codigo	Stack	Comentario
return x * fact(x-1)	fact x 1 fact x 2 fact x 3	Tu no puedes accesar la variable x desde una llamada anterior y viceversa.
if x == 1	fact x 1 fact x 2 fact x 3	
return 1	*fact* *x* *1* fact x 2 fact x 3	Este es la primer caja que es removida de la pila, lo que significa que e sla primer llamada que regresamos
return x * fact(x-1)	*fact* *x* *2* fact x 3	regresamos 2

Notemos que cada llamada tuvo su propia copia de x, tu no puedes acceder a una copia diferente de x.

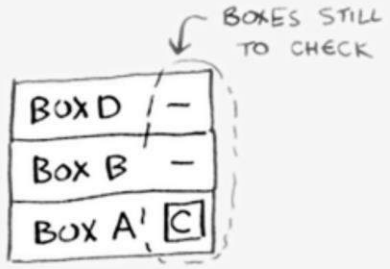
Si hacemos memoria del primer enfoque de recursión recordemos que haciamos una pila de cajas.



Pero en el enfoque recursivo no habia pila, si no hay pila como hace el algoritmo para saber que cajas todavia necesitan ser examinadas veamos el ejemplo:



En algun punto el call stack luce asi:



Ejercicio

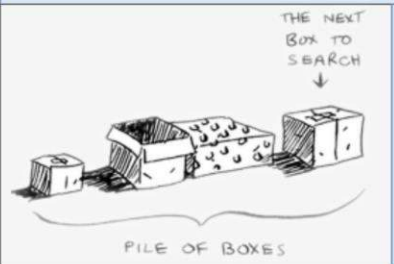
Supon que escribes una función recursiva que corre infinitamente, tu computadora guarda en memoria la dirección para cada función, ¿que pasara con el stack si tu función sigue por siempre?

Resumen

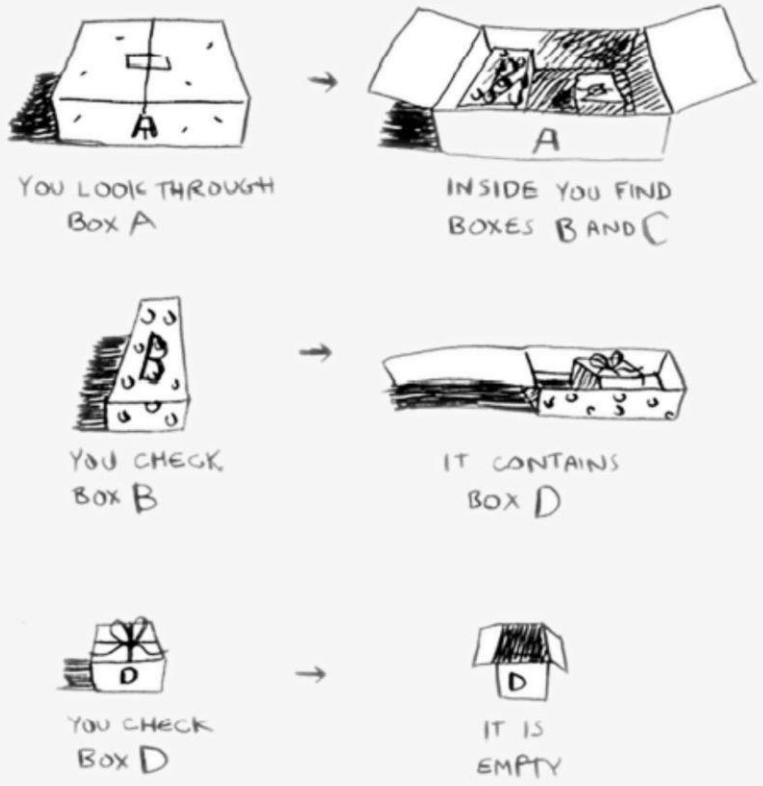
1	Recursión es cuando una **funcíon** se llama a sí **misma.**
2	Toda función recursiva tiene dos **casos**, el caso **base** y el caso **recursivo.**
3	Una pila tiene 2 operaciones **push** y **pop.**
4	Todas las llamadas de funciones van al **call stack**
5	El call stack puede ser muy grande, la cual quita mucha **memoria.**

Notemos que cada llamada tuvo su propia copia de x, tu no puedes acceder a una copia diferente de x.

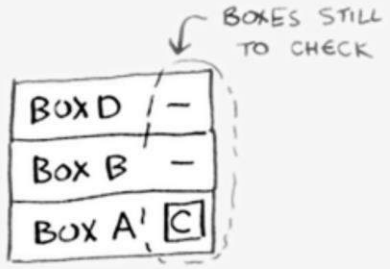
Si hacemos memoria del primer enfoque de recursión recordemos que haciamos una pila de cajas.



Pero en el enfoque recursivo no habia pila, si no hay pila como hace el algoritmo para saber que cajas todavia necesitan ser examinadas veamos el ejemplo:



En algun punto el call stack luce asi:



Ejercicio

Supon que escribes una función recursiva que corre infinitamente, tu computadora guarda en memoria la dirección para cada función, ¿que pasara con el stack si tu función sigue por siempre?

Resumen

1	Recursión es cuando una _____ se llama a sí _____.
2	Toda función recursiva tiene dos _____, el caso _____ y el caso _____.
3	Una pila tiene 2 operaciones _____ y _____.
4	Todas las llamadas de funciones van al _____ _____
5	El call stack puede ser muy grande, la cual quita mucha _____.

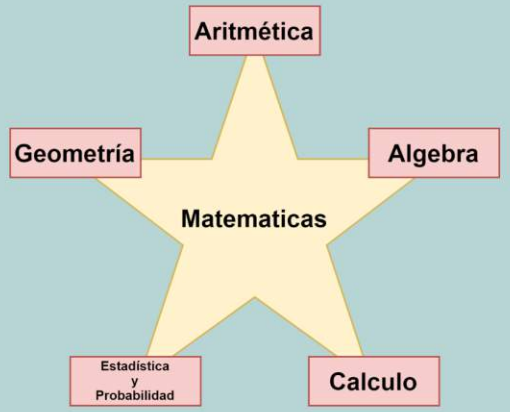
ARITMETICA

0 Matemáticas

Definición

Ciencia que estudia las propiedades y relaciones que hay entre **formas** y **abstracciones**.

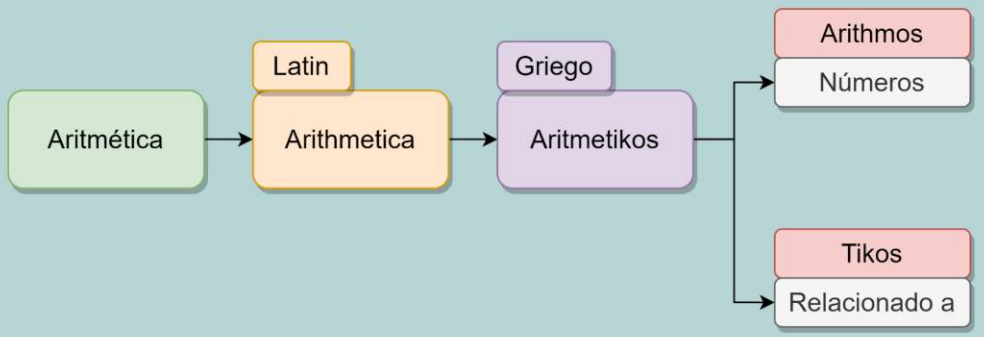
Partes



Definiciones

1	Aritmética	Área de las matemáticas que **estudia** los números enteros, raciones, reales y operaciones que **surgen** entre ellos
2	Geometría	es una rama de las matemáticas que estudia conceptos tales como **vectores**, **matrices**, sistemas de ecuaciones lineales y en su enfoque de manera más formal, espacios vectoriales y sus transformaciones lineales.
3	Algebra	es la rama de las matemáticas que estudia la **combinación** de los **números** con **letras** y símbolos y establece las reglas para manipular dichos símbolos.
4	Estadística	es una disciplina científica que se ocupa de la obtención, orden y análisis de un conjunto de **datos** con el fin de obtener explicaciones y predicciones sobre fenómenos observados.
5	Probabilidad	es el cálculo matemático que evalúa las **posibilidades** que existen de que una cosa suceda cuando interviene el **azar**.
6	Calculo	estima la magnitud de los cambios en las **variables**, así como de determinar medidas como longitudes, áreas, volúmenes, etc.

0.1 Etimología Aritmética



Definición

Aritmética viene del latín **Arithmetica**

El cual

viene del griego **aritmetikos**

A su vez

es un compuesto de otras 2 palabras griegas **Arithmos** y **tikos**.

Dando como resultado

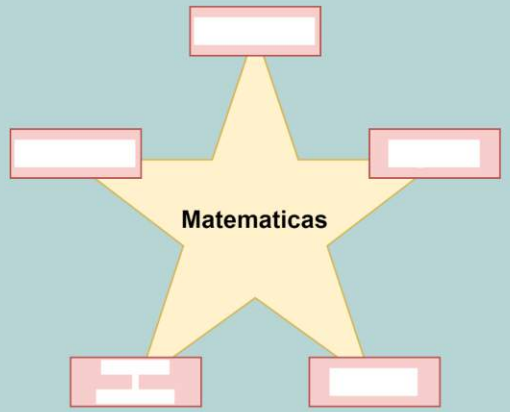
“aquello relacionado a números”

0 Matemáticas

Definición

Ciencia que estudia las propiedades y relaciones que hay entre _____ y _____.

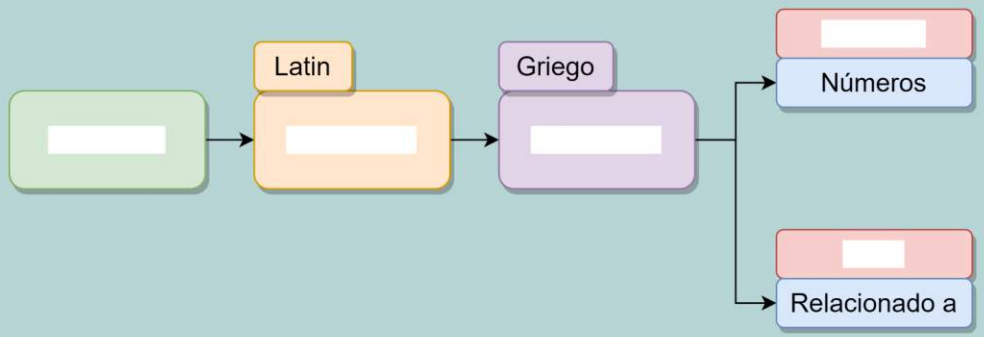
Partes



Definiciones

1	Aritmética	Área de las matemáticas que _____ los números enteros, raciones, reales y operaciones que _____ entre ellos
2	Geometría	es una rama de las matemáticas que estudia conceptos tales como _____, _____, sistemas de ecuaciones lineales y en su enfoque de manera más formal, espacios vectoriales y sus transformaciones lineales.
3	Algebra	es la rama de las matemáticas que estudia la _____ de los _____ con _____ y símbolos y establece las reglas para manipular dichos símbolos.
4	Estadística	es una disciplina científica que se ocupa de la obtención, orden y análisis de un conjunto de _____ con el fin de obtener explicaciones y predicciones sobre fenómenos observados.
5	Probabilidad	es el cálculo matemático que evalúa las _____ que existen de que una cosa suceda cuando interviene el _____.
6	Calculo	estima la magnitud de los cambios en las _____, así como de determinar medidas como longitudes, áreas, volúmenes, etc.

0.1 Etimología Aritmética



Definición

Aritmética viene del latín **Arithmetica**

El cual

viene del griego **aritmetikos**

A su vez

es un compuesto de otras 2 palabras griegas **Arithmos** y **tikos**.

Dando como resultado

“aquello relacionado a números”

Números reales

$Q \cup I$

Números racionales

Números exactos o expresiones decimales son periódicas.

Números irracionales

Números cuyas expresiones decimales no son exactas ni periódicas.

Ejemplo: π

Números enteros

Números que comprenden los naturales, negativos y el cero.

Números naturales

Conjunto formado por los números que se emplean para contar.

1 Números naturales

Definición

Conjunto formado por los números que se emplean para contar

Símbolo

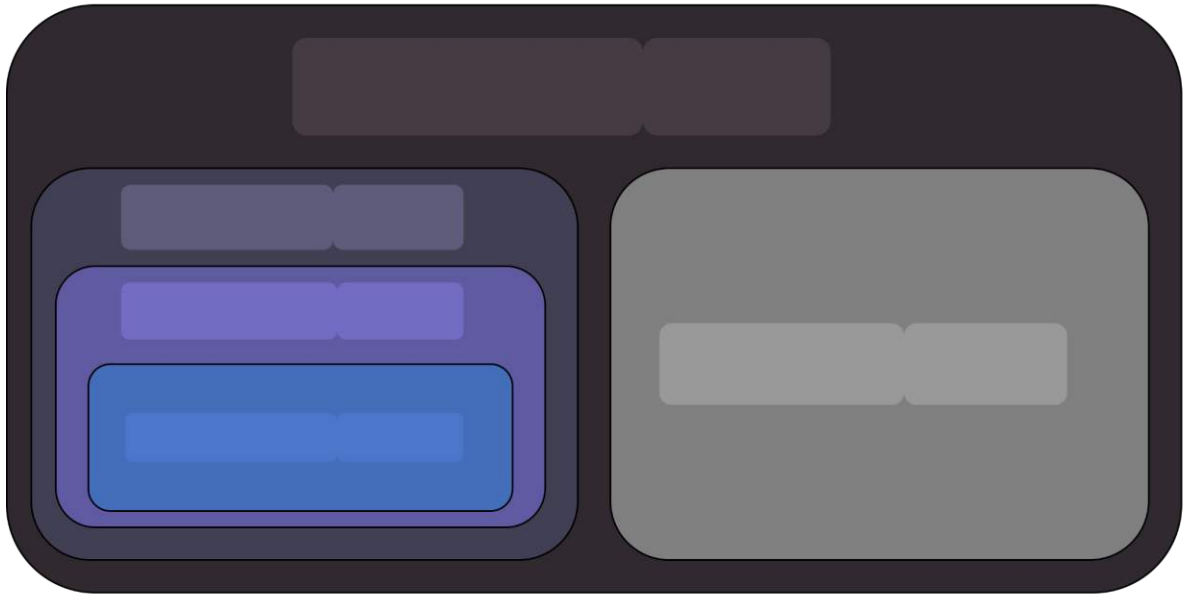
$$N$$

Ejemplo

$$N = \{1, 2, 3, 4...\}$$

Operaciones

1	Suma	+
2	Resta	-
3	Multiplicación	x
4	División	÷
5	Potenciación	x^x
6	Radicación	$\sqrt{x}$



Números reales

Números racionales

Números enteros

Números naturales

Números irracionales

Ejemplo:

1 Números naturales

Definición

Símbolo

Ejemplo

Operaciones

1 Suma

2 Resta

3 Multiplicación

4 División

5 Potenciación

6 Radicación

1.2 Operaciones

1.2.1 Suma

Símbolo +

Partes

Signo

Sumandos

Total

$$\begin{array}{r} 523 \\ +\ 172 \\ 69 \\ \hline 764 \end{array}$$

Ejercicios

$$\begin{array}{r} 53824 \\ +\ 322 \\ 1545 \\ \hline 55691 \end{array} \qquad \begin{array}{r} 65829 \\ +\ 4321 \\ 29 \\ \hline 70179 \end{array}$$

$$\begin{array}{r} 70120 \\ +\ 125 \\ 15342 \\ \hline 85587 \end{array} \qquad \begin{array}{r} 85325 \\ +\ 1002 \\ 4 \\ \hline 86331 \end{array}$$

1.2 Operaciones

1.2.2 Resta

Símbolo -

Partes

Signo

Minuendo

Sustraendo

Diferencia

$$\begin{array}{r} 714 \\ -\ 291 \\ \hline 423 \end{array}$$

Ejercicios

$$\begin{array}{r} 5290 \\ -\ 4172 \\ \hline 1118 \end{array} \qquad \begin{array}{r} 8430 \\ -\ 252 \\ \hline 8178 \end{array}$$

$$\begin{array}{r} 9432 \\ -\ 5246 \\ \hline 4186 \end{array} \qquad \begin{array}{r} 7432 \\ -\ 522 \\ \hline 6910 \end{array}$$

1.2 Operaciones

1.2.1 Suma

Símbolo

Partes

$$\begin{array}{r} 523 \\ +\ 172 \\ 69 \\ \hline 764 \end{array}$$

Ejercicios

$$\begin{array}{r} 53824 \\ 322 \\ +\ 1545 \\ \hline \end{array} \qquad \begin{array}{r} 65829 \\ +\ 4321 \\ 29 \\ \hline \end{array}$$

$$\begin{array}{r} 70120 \\ +\ 125 \\ 15342 \\ \hline \end{array} \qquad \begin{array}{r} 85325 \\ +\ 1002 \\ 4 \\ \hline \end{array}$$

1.2 Operaciones

1.2.2 Resta

Símbolo -

Partes

$$\begin{array}{r} 714 \\ -\ 291 \\ \hline 423 \end{array}$$

Ejercicios

$$\begin{array}{r} 5290 \\ -\ 4172 \\ \hline \end{array} \qquad \begin{array}{r} 8430 \\ -\ 252 \\ \hline \end{array}$$

$$\begin{array}{r} 9432 \\ -\ 5246 \\ \hline \end{array} \qquad \begin{array}{r} 7432 \\ -\ 522 \\ \hline \end{array}$$

1.2 Operaciones

1.2.3 Multiplicación

Símbolo x

Partes

signo

Multiplicando

Factores

Multiplicador

Producto

$$\begin{array}{r} 9 \\ \times\ 6 \\ \hline 54 \end{array}$$

Ejercicios

$$\begin{array}{r} 7 \\ \times\ 20 \\ \hline 140 \end{array} \qquad \begin{array}{r} 17 \\ \times\ 30 \\ \hline 510 \end{array}$$

$$\begin{array}{r} 34 \\ \times\ 16 \\ \hline 544 \end{array} \qquad \begin{array}{r} 5 \\ \times\ 4 \\ \hline 20 \end{array}$$

1.2 Operaciones

1.2.4 División

Símbolo ÷

Partes

Divisor

Cociente

Dividendo

Residuo

$$\begin{array}{r} 34 \\ 16\overline{)544} \\ 064 \\ 00 \end{array}$$

Ejercicios

$$\begin{array}{r} 3 \\ 5\overline{)15} \end{array} \qquad \begin{array}{r} 4 \\ 12\overline{)48} \end{array}$$

1.2 Operaciones

1.2.3 Multiplicación

Símbolo

Partes

$$\begin{array}{r} 9 \\ \times\ 6 \\ \hline 54 \end{array}$$

Ejercicios

$$\begin{array}{r} 7 \\ \times\ 20 \\ \hline \end{array} \qquad \begin{array}{r} 17 \\ \times\ 30 \\ \hline \end{array}$$

$$\begin{array}{r} 34 \\ \times\ 16 \\ \hline \end{array} \qquad \begin{array}{r} 5 \\ \times\ 4 \\ \hline \end{array}$$

1.2 Operaciones

1.2.4 División

Símbolo

Partes

$$\begin{array}{r} 34 \\ 16\overline{)544} \\ 064 \\ 00 \end{array}$$

Ejercicios

$$5\overline{)15} \qquad 12\overline{)48}$$

1.2 Operaciones

1.2.4 Potenciacíon

Símbolo x^x

Partes

Base — Exponente — Potencia

$$4^3 = 64$$

Ejercicios

$3^4 = 81$ $\quad$ $2^4 = 16$

$5^2 = 25$ $\quad$ $6^2 = 36$

1.2 Operaciones

1.2.5 Radicación

Símbolo $\sqrt{}$

Partes

Índice — Radical — Raíz

$$3\sqrt{64} = 4$$

Radicando

Ejercicios

$\sqrt{36} = 6$ $\quad$ $4\sqrt{81} = 3$

$5\sqrt{32} = 2$ $\quad$ $3\sqrt{125} = 5$

1.2.4 Potenciacíon

Símbolo

Partes

$$4^3 = 64$$

Ejercicios

$3^4 =$

$2^4 =$

$5^2 =$

$6^2 =$

1.2.5 Radicación

Símbolo

Partes

$$3\sqrt{64} = 4$$

Ejercicios

$\sqrt{36} =$

$4\sqrt{81} =$

$5\sqrt{32} =$

$3\sqrt{125} =$

1.3 Tipo números

Cualquier número natural puede ser de 2 tipos

1 Número primo

Definición

Son números Enteros que solo pueden dividirse entre 1 y ellos mismos.

Ejemplo

2 Divisores (2,1)
3 Divisores (3,1)

2 Número Compuesto

Definición

Son números Enteros que pueden dividirse entre 1 y más números.

Ejemplo

4 Divisores (4,2,1)
8 Divisores (8,4,2,1)

1.4 Factorización

Cualquier número natural mayor a 1 puede ser expresado de 2 maneras o número primo o producto de factores primos (3,2)

Ejemplo

24	2
12	2
6	2
3	3
1	

240	2
120	2
60	2
30	2
15	3
5	5
1	

1.5 Factorial

Definición

Es una operación de un número natural

El cual

Es el producto de todos los números naturales desde el 1 hasta el

Simbolo !

Ejemplo

1) n! = n * (n - 1)!
2) 5! = 5 * (5 - 1)!
3) 5! = 1 * 2 * 3 * 4 * 5 = 120

Ejercicios

1) 4 * 3 * 2 * 1 = ____24____ Formula = __4!__
2) 1 * 2 * 3 * 4 * 5 * 6 * 7 = ___5040___ Formula = __7!__
3) 10 * 9! = _3628800__ Formula = __10!_
4) 11! * 12 = 479001600_ Formula = __12!_

1.3 Tipo números

Cualquier número natural puede ser de 2 tipos

1 Número primo

Definición

Ejemplo

2 Divisores (,)
3 Divisores (,)

2 Número Compuesto

Definición

Ejemplo

4 Divisores (, ,)
8 Divisores (, , ,)

1.4 Factorización

Ejemplo

24	2
12	2
6	2
3	3
1	

240

1.5 Factorial

Definición

El cual

Simbolo

Ejemplo

1) n! = n * (n - 1)!
2) 5! = 5 * (5 - 1)!
3) 5! = 1 * 2 * 3 * 4 * 5 = 120

Ejercicios

1) 4 * 3 * 2 * 1 = _____ Formula = _____
2) 1 * 2 * 3 * 4 * 5 * 6 * 7 = _____ Formula = _____
3) 10 * 9! = _____ Formula = _____
4) 11! * 12 = _____ Formula = _____

ÁLGEBRA

1 Operaciones Algebraicas

*Las operaciones de gran importancia en el álgrebra y muchas mas ramas de las matemáticas son **Adición**, **sustracción**, **multiplicación**, **división**, **potenciación** y **radicación**.*

Propiedades o leyes

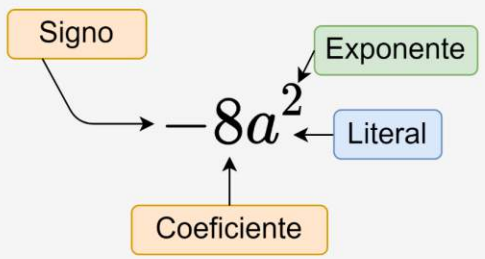
Todas las operaciones estan sujetas a restricciones llamadas **propiedades** o **leyes** y deben ser aplicadas de acuerdo a estas propiedades.

Sistema de números complejos

el sistema de números complejos es **cerrado** respecto a las 6 operaciones , es decir no se necesita más que números complejos para todos las operaciones algebraicas.

2 Expresíon Algebraica

Es cuando se contiene una o más operaciones algebraicas aplicadas a números complejos o letras cualesquiera, tambíen conocido como **termino algebraico**.



Coeficientes

Conviene considerar coeficiente solamente a un número o una letra en un termino algebraico

Ejemplo

5xy	→	**5**	→ coeficiente **númerico** de xy
bxy	→	**b**	→ coeficiente **literal** de xy

Término semejante

Es un término que difiere unicamente en sus **coeficientes**

Ejemplo

5xy **-7xy**

Grado de un término

Es la **suma** de los **exponentes** de un término algebraico

Ejemplo

$5xy$	**Grado 2**
$-2x^2$	**Grado 2**

Término racional entero

Es cuando las literales de un término algebraico estan combinadas solamente por medio de la operación de **multiplicación**.

Ejemplo

$$5xy, -3/2x^2, 5ab^2c^3$$

Suma algebraica

Si **dos** o **más** expresiones algebraicas estan enlazados por los signos + o -, la expresión se llama suma algebraica.

Ejemplo

$$4x^2 - 2x\sqrt{y} + y^2/2$$

Binomio

Es una suma algebraica de **2** terminos

Ejemplo

$$5xy + 10xy$$

Trinomio

Es una suma algebraica de **3** terminos

Ejemplo

$$5xy + 10xy + 3xy$$

Multinomio

Es una suma algebraica de **2** o **más** terminos

Ejemplo

$$4x^2 - 2x\sqrt{y} + y^2/2$$

Polinomio

Es un multinomio formado solamente por términos **racionales** enteros.

Ejemplo

$$2x^2 + 3xy + y^2$$

Grado de un polinomio

Es el grado del término de mayor grado

Ejemplo - es grado 3

$$4x^3 - 2x^2 - 8x$$

1 Operaciones Algebraicas

*Las operaciones de gran importancia en el álgrebra y muchas mas ramas de las matemáticas son **Adición**, **sustracción**, **multiplicación**, **división**, **potenciación** y **radicación**.*

Propiedades o leyes

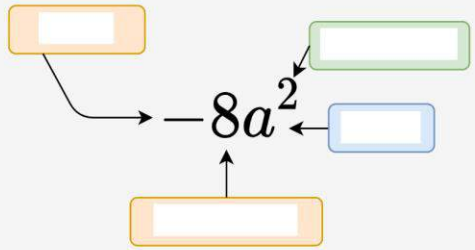
Todas las operaciones estan sujetas a restricciones llamadas _____ o _____ y deben ser aplicadas de acuerdo a estas propiedades.

Sistema de números complejos

el sistema de números complejos es _____ respecto a las 6 operaciones , es decir no se necesita más que números complejos para todos las operaciones algebraicas.

2 Expresíon Algebraica

Es cuando se contiene una o más operaciones algebraicas aplicadas a números complejos o letras cualesquiera, tambíen conocido como _____.



Coeficientes

Conviene considerar coeficiente solamente a un número o una letra en un termino algebraico

Ejemplo

5xy →	_____ →	coeficiente _____ de xy
bxy →	_____ →	coeficiente _____ de xy

Término semejante

Es un término que difiere únicamente en sus _____

Ejemplo

5xy -7xy

Grado de un término

Es la _____ de los _____ de un término algebraico

Ejemplo

$5xy$	_____
$-2x^2$	Grado 2

Término racional entero

Es cuando las literales de un término algebraico estan combinadas solamente por medio de la operación de _____.

Ejemplo

$$5xy, -3/2x^2, 5ab^2c^3$$

Suma algebraica

Si _____ o _____ expresiones algebraicas estan enlazados por los signos + o -, la expresión se llama suma algebraica.

Ejemplo

$$4x^2 - 2x\sqrt{y} + y^2/2$$

Binomio

Es una suma algebraica de _____ terminos

Ejemplo

$$5xy + 10xy$$

Trinomio

Es una suma algebraica de _____ terminos

Ejemplo

$$5xy + 10xy + 3xy$$

Multinomio

Es una suma algebraica de _____ o _____ terminos

Ejemplo

$$4x^2 - 2x\sqrt{y} + y^2/2$$

Polinomio

Es un multinomio formado solamente por términos _____ enteros.

Ejemplo

$$2x^2 + 3xy + y^2$$

Grado de un polinomio

Es el grado del término de _____ grado

Ejemplo - es grado _____

$$4x^3 - 2x^2 - 8x$$

ÁLGEBRA 2

1.2 CONCEPTOS BASICOS

Variables

Es una **letra** usada para representar varios números, y puede cambiar a travez del tiempo.

x, y , z

Constante

Es un valor númerico que no **varia**.

s = 60

Expresión algebraica

Es cualquier combinación de **números** con **letras**.

1 Conjuntos

Es una colección de objetos, que se denotan con llaves **{ }**.

$C = \{1, 2, 3\}$

Consiste

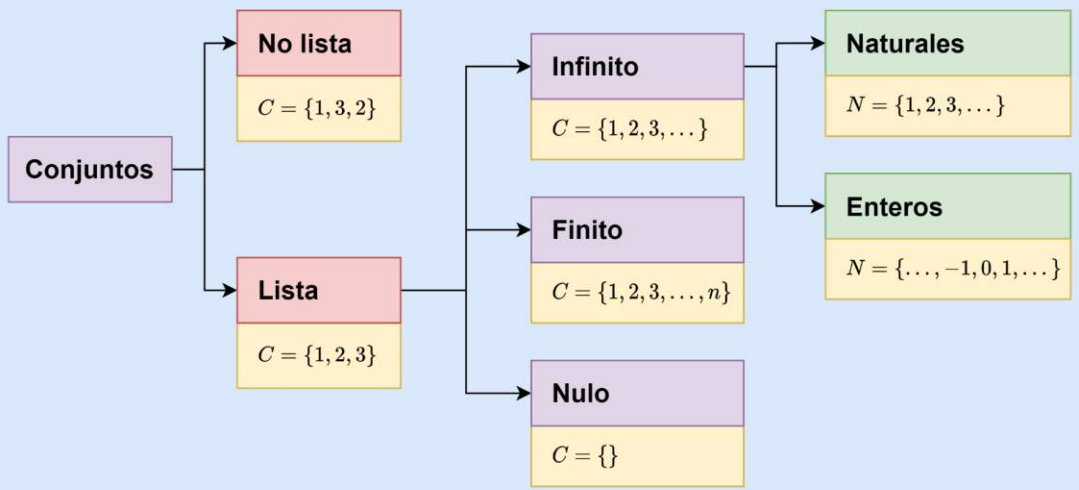
De **elementos**.

$1 \in C$

Conjunto en forma lista

Es una combinación de números y letras, **ordenado** en patrones reconocibles.

Figura 1.1



2 Desigualdades

$>$	**Es mayor que**
$\geq$	**Es mayor o igual a**
$<$	**Es menor que**
$\leq$	**Es menor o igual a**
$\neq$	**No es igual a**

Ejemplo

$6 > 2$	$-7 < 1$	$-4 > -5$

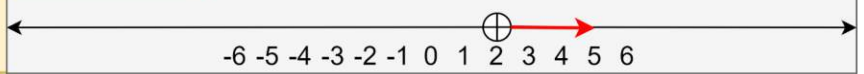
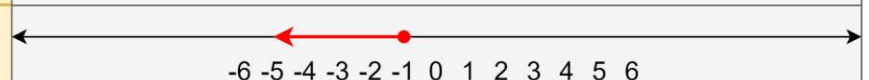
Puntos extremos

Son los extremos de un conjunto de números expresado en una desigualdad.

Ejemplo

$-4 \leq x < 3$	Los puntos extremos son **-4** y **3**.

Ejercicios

$x > 2$	
$x \leq -1$	
$-4 \leq x < 3$	

Entre e inclusive

Algunos comprenden de manera erronea la palabra *entre*

Ejercicios

Conjunto de números naturales entre 2 y 6.	$\{3, 4, 5\}$
Conjunto de números naturales entre 2 y 6 inclusive	$\{2, 3, 4, 5, 6\}$

1.2 CONCEPTOS BASICOS

Variables

Es una usada para representar varios números, y puede cambiar a travez del tiempo.

x, y , z

Constante

Es un valor númerico que no .

s = 60

Expresión algebraica

Es cualquier combinación de con .

1 Conjuntos

Es una colección de objetos, que se denotan con llaves .

$$C = \{1, 2, 3\}$$

Consiste

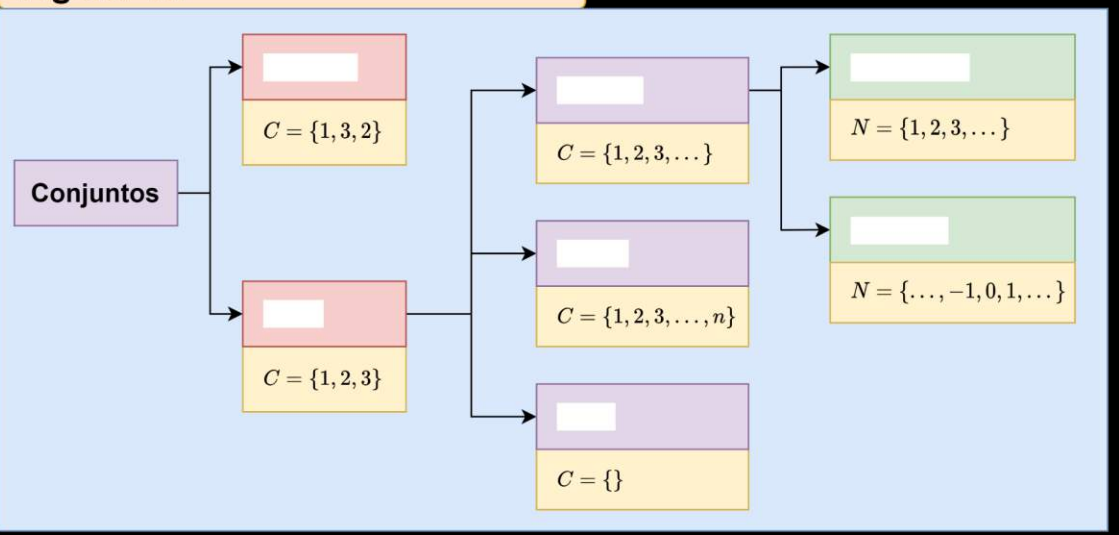
De .

$$1 \in C$$

Conjunto en forma lista

Es una combinación de números y letras, en patrones reconocibles.

Figura 1.1



2 Desigualdades

- **Es mayor que**
- **Es mayor o igual a**
- **Es menor que**
- **Es menor o igual a**
- **No es igual a**

Ejemplo

6 2	-7 1	-4 -5

Puntos extremos

Son los extremos de un conjunto de números expresado en una desigualdad.

Ejemplo

$-4 \leq x < 3$	Los puntos extremos son y .

Ejercicios

$x > 2$	-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
$x \leq -1$	-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
$-4 \leq x < 3$	-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6

Entre e inclusive

Algunos comprenden de manera erronea la palabra *entre*

Ejercicios

Conjunto de números naturales entre 2 y 6.	
Conjunto de números naturales entre 2 y 6 inclusive	

1.2 CONCEPTOS BASICOS

3 Notación constructiva de conjuntos

Es la manera matemática para expresar de forma **escrita** que elementos estan en un conjunto.

Notación

$$\{x \mid x \ \ tiene \ \ la \ \ propiedad \ \ p\}$$

El conjunto de todos los elementos x tales que

Ejemplo

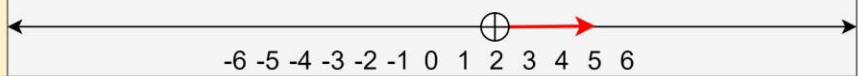
$$E = \{x | x > 7 \quad y \quad x \in N\}$$

$$E = \{8, 9, 10, 11, 12, \ldots\}$$

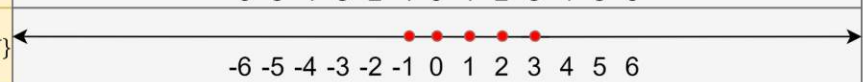
Ejercicios

$\{x|x > 2 \quad y \quad x \in N\}$

$\{x|x > 2\}$

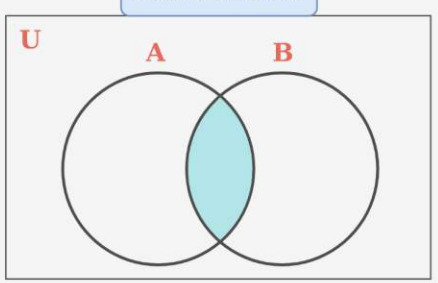


$\{x| -1 \leq x < 4 \quad y \quad x \in N\}$

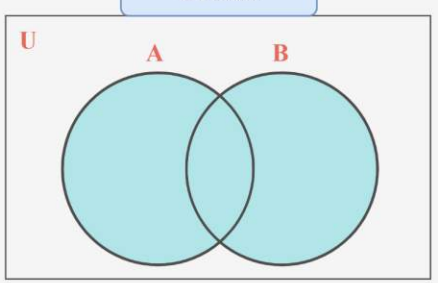


$\{x| -1 \leq x < 4\}$

Intersección



Unión



4 Unión e intersección de conjuntos

Unión e intersección son 2 ***operaciones*** que pueden hacerse a conjuntos.

A Unión

Dado un conjunto A y un conjunto B, denominamos $A \cup B$ como el conjunto de elementos que pertenecen al conjunto A ***o*** al B

Notación constructiva de conjuntos

$$A \cup B = \{x | x \in A \quad o \quad x \in B\}$$

Ejercicio

$A = \{1, 2, 3, 4, 5\}$	$B = \{3, 4, 5, 6, 7\}$	$A \cup B = \{1, 2, 3, 4, 5, 6, 7\}$
$A = \{a, b, c, d, e\}$	$B = \{x, y, z\}$	$A \cup B = \{a, b, c, d, e, x, y, z\}$

B Intersección

Dado un conjunto A y un conjunto B, denominamos $A \cap B$ como el conjunto de elementos que pertenecen al conjunto A ***y*** al B

Notación constructiva de conjuntos

$$A \cap B = \{x | x \in A \quad y \quad x \in B\}$$

Ejercicio

$A = \{1, 2, 3, 4, 5\}$	$B = \{3, 4, 5, 6, 7\}$	$A \cap B = \{3, 4, 5\}$
$A = \{a, b, c, d, e\}$	$B = \{x, y, z\}$	$A \cap B = \{\}$

1.2 CONCEPTOS BASICOS

3 Notación constructiva de conjuntos

Es la manera matemática para expresar de forma _____ que elementos estan en un conjunto.

Notación

$$\{x \mid x \text{ tiene la propiedad } p\}$$

_____ _____ _____

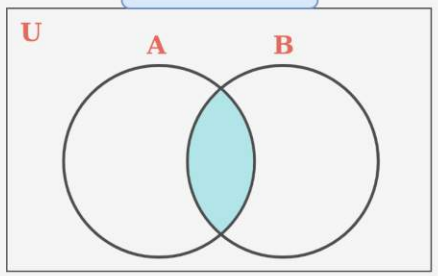
Ejemplo

$$E = \{x | x > 7 \quad y \quad x \in N\}$$

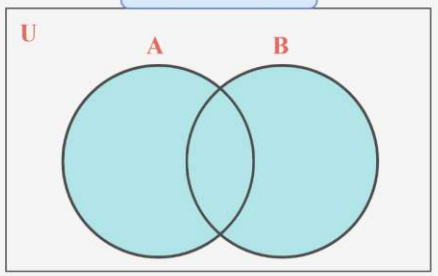
Ejercicios

$\{x\|x > 2 \quad y \quad x \in N\}$	-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
$\{x\|x > 2\}$	-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
$\{x\| -1 \leq x < 4 \quad y \quad x \in N\}$	-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
$\{x\| -1 \leq x < 4\}$	-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6

_____



_____



4 Unión e intersección de conjuntos

Unión e intersección son 2 _____ que pueden hacerse a conjuntos.

A Unión

Dado un conjunto A y un conjunto B, denominamos $A \cup B$ como el conjunto de elementos que pertenecen al conjunto A _____ al B

Notación constructiva de conjuntos

$$A \cup B =$$

Ejercicio

$A = \{1, 2, 3, 4, 5\}$	$B = \{3, 4, 5, 6, 7\}$	$A \cup B = \{1, 2, 3, 4, 5, 6, 7\}$
$A = \{a, b, c, d, e\}$	$B = \{x, y, z\}$	$A \cup B = \{a, b, c, d, e, x, y, z\}$

B Intersección

Dado un conjunto A y un conjunto B, denominamos $A \cap B$ como el conjunto de elementos que pertenecen al conjunto A _____ al B

Notación constructiva de conjuntos

$$A \cap B =$$

Ejercicio

$A = \{1, 2, 3, 4, 5\}$	$B = \{3, 4, 5, 6, 7\}$	$A \cap B =$
$A = \{a, b, c, d, e\}$	$B = \{x, y, z\}$	$A \cap B =$

1.2 CONCEPTOS BASICOS

5 Conjunto importantes de numeros

Existen **conjuntos** de números reales que es necesario estudiar.

Conjuntos importantes

1	**Números reales**	R = {x|x es un punto de la recta númerica}
2	**Números racionales**	$Q = \{\frac{p}{q}, p, q \in Z, q \neq 0\}$
3	**Números irracionales**	I = {x|x es número real que no es racional}
4	**Números enteros**	$Z = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
5	**Números naturales**	$N = \{1, 2, 3, 4, 5, \ldots\}$

A Número racional

Cualquier número que puede representarse como un **cociente** de 2 enteros con el denominador distinto de 0.

Ejemplo	$\frac{3}{5}, -\frac{2}{3}, \frac{0}{1}$

Racional decimal

Todo número racional escrito en decimal será un número con parte decimal que se **repite** o bien que **termina**.

Ejemplo	$\frac{2}{3} = 0.666... \quad , \frac{1}{7} = 0.142857142857...$

Podemos poner una **barra** sobre el dígito para representar su repetición

Ejemplo	$\frac{2}{3} = 0.\overline{6} \quad , \frac{1}{7} = 0.\overline{142857}$

B Número irracional

Número que no se **repite** ni **termina** su parte decimal, la mayoria de las **raíces** cuadradas son irracionales.

Ejemplo	$\pi, \sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{6}$

Para representar que valor se **aproxima** al valor del número irracional usamos:

$\approx$	**"es aproximadamente igual a"**

Ejemplo

$\pi \approx 3.14$	$\sqrt{2} \approx 1.41$	$\sqrt{3} \approx 1.73$	$\sqrt{10} \approx 3.16$

C Números reales

Son formados por la **unión** de los números racionales y los números irracaionales

R

Ejercicio

$$\{-3, 0, \frac{5}{7}, 12.25, \sqrt{7}, -\sqrt{11}, \frac{22}{7}, 5, 7.1, -54, \pi\}$$

1	**Números naturales**	5
2	**Números racionales**	$-3, 0, \frac{5}{7}, 12.25, -\frac{22}{7}, 5, 7.1, -54$
3	**Enteros no negativos**	0 , 5
4	**Números irracionales**	$\sqrt{7}, -\sqrt{11}, \pi$
5	**Enteros**	-3, 0, 5, -54
6	**Números reales**	Todos

1.2 CONCEPTOS BASICOS

5 Conjunto importantes de numeros

Existen _____ de números reales que es necesario estudiar.

Conjuntos importantes

1	Números reales	R = {x|x es un punto de la recta númerica}
2	Números racionales	$Q = \{\frac{p}{q}, p, q \in Z, q \neq 0\}$
3	Números irracionales	I = {x|x es número real que no es racional}
4	Números enteros	
5	Números naturales	

A Número racional

Cualquier número que puede representarse como un _____ de 2 enteros con el denominador distinto de 0.

Ejemplo	$\frac{3}{5}, -\frac{2}{3}, \frac{0}{1}$

Racional decimal

Todo número racional escrito en decimal será un número con parte decimal que se _____ o bien que _____.

Ejemplo	$\frac{2}{3} = 0.666...$, $\frac{1}{7} = 0.142857142857...$

Podemos poner una _____ sobre el dígito para representar su repetición

Ejemplo	$\frac{2}{3} =$ _____ , $\frac{1}{7} =$ _____

B Número irracional

Número que no se _____ ni _____ su parte decimal, la mayoria de las _____ cuadradas son irracionales.

Ejemplo	$\pi, \sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{6}$

Para representar que valor se _____ al valor del número irracional usamos:

$\approx$	" _____ "

Ejemplo

π _____ 3.14	$\sqrt{2}$ _____ 1.41	$\sqrt{3}$ _____ 1.73	$\sqrt{10}$ _____ 3.16

C Números reales

Son formados por la _____ de los números racionales y los números irracaionales

R

Ejercicio

$$\{-3, 0, \frac{5}{7}, 12.25, \sqrt{7}, -\sqrt{11}, \frac{22}{7}, 5, 7.1, -54, \pi\}$$

1	Números naturales	
2	Números racionales	
3	Enteros no negativos	
4	Números irracionales	
5	Enteros	
6	Números reales	

1.2 CONCEPTOS BASICOS

EJERCICIOS 1

1	Todo número natural es un entero no negativo.	V
2	Todo entero no negativo es un número natural.	V
3	Algunos números racionales son enteros.	F
4	Todo entero es un número racional	V
5	Todo número racional es un entero.	F
6	La unión del conjunto de los números racionales con el conjunto de los números irracionales forma el conjunto de los números reales.	V
7	La intersección del conjunto de los números racionales y el conjunto de los números irracionales forma el conjunto de los números reales.	F
8	El conjunto de los números naturales es el conjunto vacío.	F
9	El conjunto de los enteros entre (π y 4) es el conjunto vacío(nulo).	V
10	El conjunto de los números irracionales entre 3 y π es un conjunto infinito.	F

EJERCICIOS 2

5 > 3	-1 < 8	0 > -2	-3 < 3
-1 > -1.01	2 > -3	-5 < -3	-8 < -1
-14.98 > -14.99	-3.4 < -3.2	1.7 < 1.9	-1.1 > -1.9
π > -4	-723 < -655	-7/8 > -10/11	-4/7 < -5/9

EJERCICIOS 3

1	A = {x|-1 < x < 1 y x $\in Z$}	$\{-1, 0, 1\}$
2	B = {y|y es un número natural impar menor que 6}	$\{1, 3, 5\}$
3	C = {z|z es un entero par mayor que 16 y menor o igual a 20}	$\{18, 20\}$
4	D = {x|x $\leq$ -3 y x $\in$ I}	$\{-4\}$
5	L = {x|x es un entero mayor que -5}	$\{-4\}$

EJERCICIOS 4

A	B	$A \cup B$	$A \cap B$
$\{1, 2, 3\}$	$\{4, 5, 6\}$	$\{1, 2, 3, 4, 5, 6\}$	$\{\}$
$\{\}$	$\{2, 4, 6, 8, 10\}$	$\{2, 4, 6, 8, 10\}$	$\{\}$
$\{1, 2, 3, 4, 5\}$	$\{2, 4, 6\}$	$\{1, 2, 3, 4, 5, 6\}$	$\{2, 4\}$

EJERCICIOS 5

$B = \{2, 4, 6, 8\}$	
Conjunto múltiplo de 2	
$B = \{1, 2, 3, 4, \ldots\}$	
Conjunto de números naturales	
$B = \{0, 3, 6, 9, \ldots\}$	
Conjunto múltiplo de 3	
$B = \{a, b, c, d, \ldots, z\}$	
Conjunto abecedario	

1.2

CONCEPTOS BASICOS

EJERCICIOS 1

1	Todo número natural es un entero no negativo.	
2	Todo entero no negativo es un número natural.	
3	Algunos números racionales son enteros.	
4	Todo entero es un número racional	
5	Todo número racional es un entero.	
6	La unión del conjunto de los números racionales con el conjunto de los números irracionales forma el conjunto de los números reales.	
7	La intersección del conjunto de los números racionales y el conjunto de los números irracionales forma el conjunto de los números reales.	
8	El conjunto de los números naturales es el conjunto vacío.	
9	El conjunto de los enteros entre (π y 4) es el conjunto vacío(nulo).	
10	El conjunto de los números irracionales entre 3 y π es un conjunto infinito.	

EJERCICIOS 2

5 3	-1 8	0 -2	-3 3
-1 -1.01	2 -3	-5 -3	-8 -1
-14.98 -14.99	-3.4 -3.2	1.7 1.9	-1.1 -1.9
π -4	-723 -655	-7/8 -10/11	-4/7 -5/9

EJERCICIOS 3

1	A = {x|-1 < x < 1 y x $\in Z$}	
2	B = {y|y es un número natural impar menor que 6}	
3	C = {z|z es un entero par mayor que 16 y menor o igual a 20}	
4	D = {x|x $\leq$ -3 y x $\in$ I}	
5	L = {x|x es un entero mayor que -5}	

EJERCICIOS 4

A	B	$A \cup B$	$A \cap B$
$\{1,2,3\}$	$\{4,5,6\}$		
$\{\}$	$\{2,4,6,8,10\}$		
$\{1,2,3,4,5\}$	$\{2,4,6\}$		

EJERCICIOS 5

$B = \{2,4,6,8\}$	
$B = \{1,2,3,4,\ldots\}$	
$B = \{0,3,6,9,\ldots\}$	
$B = \{a,b,c,d,\ldots,z\}$	

1.3 Propiedades y operaciones con los números reales

Introducción

Antes de explicar la suma y resta de números reales necesitamos estudiar el valor absoluto.

¿Qué es un inverso aditivo?

Son dos números en la recta númerica, que estan a la misma distancia pero en direcciones opuestas.

Definición de inverso aditivo

Para cualquier número real a, su inverso aditivo es $-a$.

Ejemplo

1) Inverso aditivo de -3 es 3.
2) Inverso aditivo de 3 es -3.
3) Inverso aditivo de 0 es 0.
4) Inverso aditivio de -56.3 es 56.3.

Propiedad del doble negativo

Para cualquier número real a, $-(-a) = a$.

Ejemplos

1) $-(-7.4) = 7.4$
2) $-(-\frac{12}{5}) = \frac{12}{5}$

¿Qué es el valor absoluto?

El valor absoluto de un número es su distancia, con respecto al 0, en la recta númerica.

Como se denota

El símbolo || se usa para denotar el valor absoluto.

Definición

Si a es un número real, entonces.

$$|a| \begin{cases} a \text{ si } a \geq 0 \\ -a \text{ si } a < 0 \end{cases}$$

Ejercicio

Evalue los siguientes valores absolutos.

1	$\lvert 7\rvert = 7$	**5**	$-\lvert 5\rvert = -5$
2	$\lvert -8.2\rvert = 8.2$	**6**	$-\lvert -6.43\rvert = -6.43$
3	$\lvert 0\rvert = 0$	**7**	$\lvert -12\rvert = 12$
4	$\lvert 6.3\rvert = 6.3$		

Evalue los siguientes valores absolutos.

1	$\lvert 8\rvert = \lvert -8\rvert$	**5**	$\lvert -1\rvert > -\lvert -3\rvert$

1.3 Propiedades y operaciones con los números reales

Introducción

Antes de explicar la suma y resta de números reales necesitamos estudiar el valor absoluto.

¿Qué es un inverso aditivo?

Son dos números en la recta númerica, que estan a la misma pero en direcciones opuestas.

Definición de inverso aditivo

Para cualquier número real a, su inverso aditivo es $-a$.

Ejemplo

1) Inverso aditivo de -3 es .

2) Inverso aditivo de 3 es .

3) Inverso aditivo de 0 es .

4) Inverso aditivio de -56.3 es .

Propiedad del doble negativo

Para cualquier número real a, $-(-a) = a$.

Ejemplos

1) $-(-7.4) =$

2) $-(-\frac{12}{5}) =$

¿Qué es el valor absoluto?

El valor absoluto de un número es su distancia, con respecto al 0, en la recta númerica.

Como se denota

El símbolo se usa para denotar el valor absoluto.

Definición

Si a es un número real, entonces.

$$|a| \begin{cases} a \text{ si } \quad 0 \\ -a \text{ si } \quad 0 \end{cases}$$

Ejercicio

Evalue los siguientes valores absolutos.

1	$\lvert 7\rvert =$	5	$-\lvert 5\rvert =$
2	$\lvert -8.2\rvert =$	6	$-\lvert -6.43\rvert =$
3	$\lvert 0\rvert =$	7	$\lvert -12\rvert =$
4	$\lvert 6.3\rvert =$		

Evalue los siguientes valores absolutos.

1	$\lvert 8\rvert \quad \lvert -8\rvert$	5	$\lvert -1\rvert \quad -\lvert -3\rvert$

6 Usar las propiedades de los números reales

A continuación listaremos las propiedades básicas para las operaciones de suma y multiplicación de números reales.

Para números reales a,b y c	Suma	Multiplicación
Propiedad conmutativa	$a+b=b+a$	$ab=ba$
Propiedad asociativa	$(a+b)+c=a+(b+c)$	$(ab)c=a(bc)$
Propiedad de la identidad	$a+0=0+a=a$ 0 se elimina **elemento idéntico aditivo**	$a*1=1*a=a$ 1 se denomina **elemento idéntico multiplicativo**
Propiedad del inverso	$a+(-a)=(-a)+a=0$ $-a$ se denomina **inverso aditivo** u **opuesto** de a	$a*\frac{1}{a}=\frac{1}{a}*a=1$ $\frac{1}{a}$ se denomina **inverso multiplicativo** o **recíproco** de $a, a\neq 0$
Propiedad distributiva	$a(b+c)=ab+ac$	

Observación

Propiedad conmutativa	Esta implica un cambio en el orden
Propiedad asociativa	Esta implica un cambio en la agrupación
Propiedad distributiva	Se aplica cuando hay más de dos números dentro de los paréntesis.

$$a(b+c+d+\ldots+n)=ab+ac+ad+\ldots+an$$

Diga el nombre de la propiedad

$7*m=m*7$	Propiedad conmutativa de la multiplicación
$4s+5t=5t+4s$	Propiedad conmutativa de la suma
$2v(w+3)=2v*w+2v*3$	Propiedad distributiva
$(a+8)+2b=a+(8+2b)$	Propiedad asociativa de la suma

Diga el nombre de la propiedad

$9*1=9$	Propiedad del idéntico multiplicativo
$4+(-4)=0$	Propiedad del inverso aditivo
$x+0=x$	Propiedad del idéntico aditivo
$1(x+y)=x+y$	Propiedad del idéntico multiplicativo
$b*0=0$	Propiedad del cero en la multiplicación
$x=1*x$	Propiedad del idéntico multiplicativo
$4*\frac{1}{4}=1$	Propiedad del inverso multiplicativo
$4(x+y+2)=4x+4y+8$	Propiedad distributiva

Escribe el inverso aditivo u opuesto y el inverso multiplicativo o recíproco de cada uno de los siguientes

-3	El inverso aditivio es 3	El inverso multiplicativo es $\frac{1}{-3}=-\frac{1}{3}$
$\frac{2}{3}$	El inverso aditivo es $-\frac{2}{3}$	El inverso multiplicativo es $\frac{1}{\frac{2}{3}}=\frac{2}{3}$
6	El inverso aditivo es -6	El inverso multiplicativo es $-\frac{1}{6}$
-13	El inverso aditivo es -13	El inverso multiplicativo es $-\frac{1}{13}$
$-\frac{22}{7}$	El inverso aditivo es $\frac{22}{7}$	El inverso multiplicativo es $\frac{1}{\frac{22}{7}}$
$-\frac{3}{5}$	El inverso aditivo es $\frac{3}{5}$	El inverso multiplicativo es $\frac{1}{\frac{3}{5}}$

6 Usar las propiedades de los números reales

A continuación listaremos las propiedades básicas para las operaciones de suma y multiplicación de números reales.

Para números reales a,b y c	Suma	Multiplicación
Propiedad conmutativa	$a+b=b+a$	$ab=ba$
Propiedad asociativa	$(a+b)+c=a+(b+c)$	$(ab)c=a(bc)$
Propiedad de la identidad	$a+0=0+a=a$ 0 se elimina **elemento idéntico aditivo**	$a*1=1*a=a$ 1 se denomina **elemento idéntico multiplicativo**
Propiedad del inverso	$a+(-a)=(-a)+a=0$ $-a$ se denomina **inverso aditivo** u **opuesto** de a	$a*\frac{1}{a}=\frac{1}{a}*a=1$ $\frac{1}{a}$ se denomina **inverso multiplicativo** o **recíproco** de $a, a\neq 0$
Propiedad distributiva	$a(b+c)=ab+ac$	

Observación

Propiedad conmutativa	Esta implica un cambio en el orden
Propiedad asociativa	Esta implica un cambio en la agrupación
Propiedad distributiva	Se aplica cuando hay más de dos números dentro de los paréntesis.

$$a(b+c+d+\ldots+n)=ab+ac+ad+\ldots+an$$

Diga el nombre de la propiedad

$7*m=m*7$	Propiedad
$4s+5t=5t+4s$	Propiedad
$2v(w+3)=2v*w+2v*3$	Propiedad
$(a+8)+2b=a+(8+2b)$	Propiedad

Diga el nombre de la propiedad

$9*1=9$	Propiedad del
$4+(-4)=0$	Propiedad del
$x+0=x$	Propiedad del
$1(x+y)=x+y$	Propiedad del
$b*0=0$	Propiedad del
$x=1*x$	Propiedad del
$4*\frac{1}{4}=1$	Propiedad del
$4(x+y+2)=4x+4y+8$	Propiedad

Escribe el inverso aditivo u opuesto y el inverso multiplicativo o recíproco de cada uno de los siguientes

-3	El inverso aditivio es	El inverso multiplicativo es $=-\frac{1}{3}$
$\frac{2}{3}$	El inverso aditivo es	El inverso multiplicativo es $=\frac{2}{3}$
6	El inverso aditivo es	El inverso multiplicativo es
-13	El inverso aditivo es	El inverso multiplicativo es
$-\frac{22}{7}$	El inverso aditivo es	El inverso multiplicativo es
$-\frac{3}{5}$	El inverso aditivo es	El inverso multiplicativo es

3 Evaluar expresiones por medio del orden de las operaciones

Con frecuencia se tendra que evaluar expresiones que tienen varias operaciones, para haceer esto se tiene que seguir un orden o jerarquía.

Orden de las operaciones

1	Primero, se evalua las expresiones dentro de símbolos de agrupación como (), [], {} y ||. Si esta esta anidada, evalúe las expresiones más intensas.
2	Después, evalue todos los terminos que tengan exponentes y raíces.
3	A continuación, evalue multiplicaciones y divisiones, en su orden, de izquierda a derecha.
4	Por último evalue sumas y restas, en su orden, de izquierda a derecha.

Evalue $6 + 3 * 5^2 - 10$

$$6 + 3 * 5^2 - 10 = 6 + 3 * 25 - 10 = 6 + 75 - 10 = 81 - 10 = 71$$

Evalue $10 + \{6 - [4(5-2)]\}^2$

$$10 + \{6 - [4(3)]\}^2 = 10 + \{6 - [12]\}^2 = 10 + \{-6\}^2 = 10 + 36 = 46$$

1.5 Exponentes

1 Usar la regla del producto para exponentes

Considera esta multiplicación $x^3 * x^5$ podemos simplificarla como sigue:

$$x^3 * x^5 = (x * x * x) * (x * x * x * x * x) = x^8$$

Este problema podria simplificarse por medio de la regla del producto para exponentes:

regla del producto para exponentes

Si m y n son números naturales y a es cualquier número natural real, entonces:

$$a^m * a^n = a^{m+n}$$

Evalue

a	$2^3 * 2^4 = 2^7 = 128$
b	$d^2 * d^5 = d^7$
c	$h * h^9 = h^{10}$

2 Cociente para exponentes

Considera la división $\frac{x^7}{x^4}$ podemos simplificar esta expresión como sigue:

$$\frac{x^7}{x^4} = \frac{x * x * x * x * x * x * x}{x * x * x * x} = x * x * x = x^3$$

Este problema podria simplificarse por medio de la regla del cociente para exponentes:

regla del cociente para exponentes

Si a es cualquier número real diferente de 0, y m y n son enteros diferentes de 0, entonces:

$$\frac{a^m}{a^n} = a^{m-n}$$

Evalue

a	$\frac{6^4}{6^2} = 6^{4-2} = 6^2 = 36$	b	$\frac{x^7}{x^3} = x^{7-3} = x^4$
c	$\frac{y^2}{y^5} = y^{2-5} = y^{-3}$		

3 Evaluar expresiones por medio del orden de las operaciones

Con frecuencia se tendra que evaluar expresiones que tienen varias operaciones, para haceer esto se tiene que seguir un orden o jerarquía.

Orden de las operaciones

1	Primero, se evalua las expresiones dentro de símbolos de agrupación como (), [], {} y ||. Si esta esta anidada, evalúe las expresiones más intensas.
2	Después, evalue todos los terminos que tengan exponentes y raíces.
3	A continuación, evalue multiplicaciones y divisiones, en su orden, de izquierda a derecha.
4	Por último evalue sumas y restas, en su orden, de izquierda a derecha.

Evalue $6 + 3 * 5^2 - 10$

$6 + 3 * 5^2 - 10 = \quad = \quad = \quad =$

Evalue $10 + \{6 - [4(5-2)]\}^2$

$10 + \{6 - [4(3)]\}^2 = \quad = \quad = \quad =$

1.5 Exponentes

1 Usar la regla del producto para exponentes

Considera esta multiplicación $x^3 * x^5$ podemos simplificarla como sigue:

$$x^3 * x^5 = (x * x * x) * (x * x * x * x * x) = x^8$$

Este problema podria simplificarse por medio de la regla del producto para exponentes:

regla del producto para exponentes

Si m y n son números naturales y a es cualquier número natural real, entonces:

$$a^m * a^n = \quad$$

Evalue

a	$2^3 * 2^4 = \quad = 128$
b	$d^2 * d^5 = \quad$
c	$h * h^9 = \quad$

2 Cociente para exponentes

Considera la división $\frac{x^7}{x^4}$ podemos simplificar esta expresión como sigue:

$$\frac{x^7}{x^4} = \frac{x * x * x * x * x * x * x}{x * x * x * x} = x * x * x = \quad$$

Este problema podria simplificarse por medio de la regla del cociente para exponentes:

regla del cociente para exponentes

Si a es cualquier número real diferente de 0, y m y n son enteros diferentes de 0, entonces:

$$\frac{a^m}{a^n} = \quad$$

Evalue

a	$\frac{6^4}{6^2} = \quad = \quad = 36$	b	$\frac{x^7}{x^3} = \quad = \quad$
c	$\frac{y^2}{y^5} = \quad = \quad$		

3 Usar la regla del exponente negativo

Algunos exponentes pueden ser negativos por ejemplo $\frac{y^2}{y^5} = y^{2-5} = y^{-3}$, si cancelamos factores comunes tenemos:

$$\frac{y^2}{y^5} = \frac{y * y}{y * y * y * y * y} = \frac{1}{y^3}$$

Al reducir factores comunes podemos razonar que $y^{-3} = \frac{1}{y^3}$, este es la regla del exponente negativo.

Regla del exponente negativo

Para cualquier número real diferente de 0, a y cualquier número entero no negativo, m.

$$a^{-m} = \frac{1}{a^m}$$

Escriba cada expresión sin exponentes negativos

1	$7^{-2} = \frac{1}{7^2} = \frac{1}{49}$
2	$8a^{-4} = 8 * \frac{1}{a^4} = \frac{8}{a^4}$
3	$\frac{1}{C^{-5}} = 1 \div C^{-5} = 1 \div \frac{1}{C^5} = \frac{1}{1} * \frac{C^5}{1} = C^5$

En general, nunca dejamos expresiones exponenciales con exponentes negativos.

3 Usar la regla del exponente 0

Cualquier número distinto de cero dividido entre si mismo es 1, Por lo tanto:

$$\frac{x^5}{x^5} = 1$$

Y por medio de la regla del cociente tenemos

$$\frac{x^5}{x^5} = x^{5-5} = x^0$$

Entonces $x^0 = 1$

Regla del exponente 0

Si a es cualquier número real diferente de 0, entonces:

$$a^0 = 1$$

Evalue

1	$162^0 = 1$
2	$7^0 = 7 * p^0 = 7 * 1 = 7$
3	$-y^0 = -1 * y^0 = -1 * 1 = -1$
4	$-(8x + 9y)^0 = -1 * (8x + 9y)^0 = -1 * 1 = -1$

3 Usar la regla del exponente negativo

Algunos exponentes pueden ser negativos por ejemplo $\frac{y^2}{y^5} = y^{2-5} = y^{-3}$, si cancelamos factores comunes tenemos:

$$\frac{y^2}{y^5} = \frac{y * y}{y * y * y * y * y} = \square$$

Al reducir factores comunes podemos razonar que $y^{-3} = \square$, este es la regla del exponente negativo.

Regla del exponente negativo

Para cualquier número real diferente de 0, a y cualquier número entero no negativo, m.

$$a^{-m} = \square$$

Escriba cada expresión sin exponentes negativos

1	$7^{-2} = \square = \square$
2	$8a^{-4} = \square \cdot = \square$
3	$\frac{1}{C^{-5}} = \square = \square = \square = \square$

En general, nunca dejamos expresiones exponenciales con exponentes negativos.

3 Usar la regla del exponente 0

Cualquier número distinto de cero dividido entre si mismo es 1, Por lo tanto:

$$\frac{x^5}{x^5} = \square$$

Y por medio de la regla del cociente tenemos

$$\frac{x^5}{x^5} = x^{5-5} = \square$$

Entonces $x^0 = \square$

Regla del exponente 0

Si a es cualquier número real diferente de 0, entonces:

$$a^0 = \square$$

Evalue

1	$162^0 = \square$
2	$7^0 = \square = \square = \square$
3	$-y^0 = \square = \square = \square$
4	$-(8x + 9y)^0 = \square = \square = \square$

Investigación de operaciones

1 Introducción

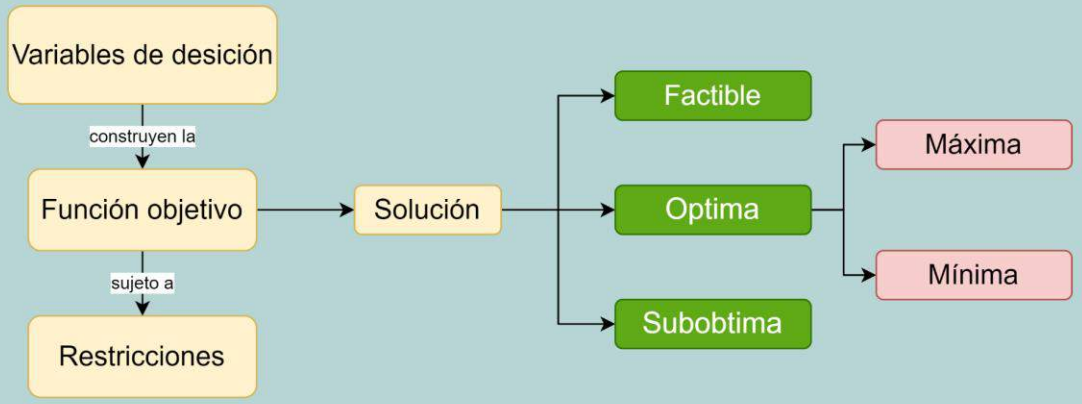
History

*Las primeras actividades de **investigación de operaciones** se dieron durante la **segunda guerra mundial**, cuando se encomendó a un equipo de **científicos ingleses** la toma de decisiones acerca de la mejor utilización de materiales bélicos.*

*Las ideas formuladas en operaciones militares fueron adaptadas al **sector civil.***

*Un elemento principal de la investigación de operaciones es el **modelado matemático**.*

1.1 Modelos de investigación de operaciones



Definiciones

*Los componentes principales del modelado son **variables**, **función objetivo**, y **restricciones**.*

El primer paso

*de cualquier **modelo matemático** es la definición de las **variables** de **decisión** del problema.*

A continuación

*se usan las **variables de desición** para construir la **función objetivo** y las **restricciones** del **modelo**.*

Una solución factible

*es factible si satisface todas las **restricciones***

Es óptima si

*además de ser **factible**, produce el **mejor** valor (**máximo** o **mínimo**) de la función objetivo.*

1 Introducción

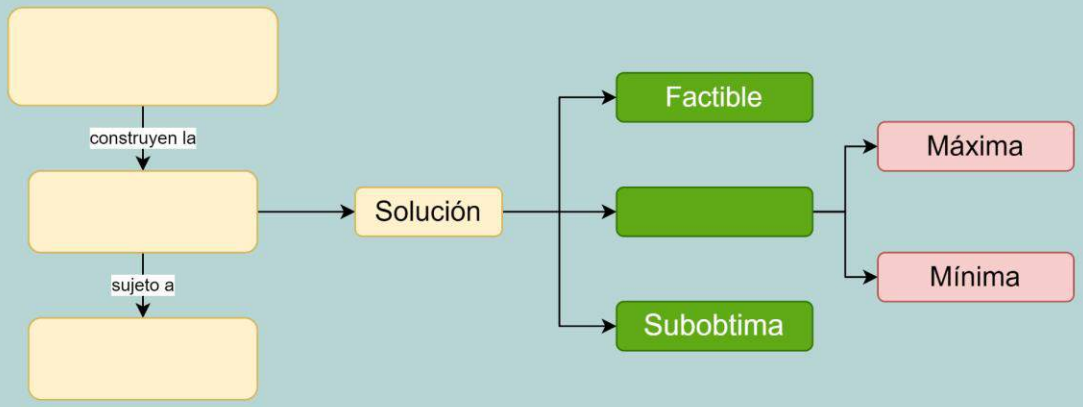
History

Las primeras actividades de ____ ***de*** ____ *se dieron durante la* ____ *, cuando se encomendó a un equipo de* ____ *la toma de decisiones acerca de la mejor utilización de materiales bélicos.*

Las ideas formuladas en operaciones militares fueron adaptadas al ____ ***.***

Un elemento principal de la investigación de operaciones es el ____ *.*

1.1 Modelos de investigación de operaciones



Definiciones

Los componentes principales del modelado son ____ *,* ____ *, y* ____ *.*

El primer paso

de cualquier ____ *es la definición de las* ____ *de* ____ *del problema.*

A continuación

se usan las ____ *para construir la* ____ *y las* ____ *del* ____ *.*

Una solución factible

es factible si satisface todas las ____

Es óptima si

además de ser ____ *, produce el* ***mejor*** *valor (* ____ *o* ____ *) de la función objetivo.*

1.2 Solución del modelo de investigación de operaciones

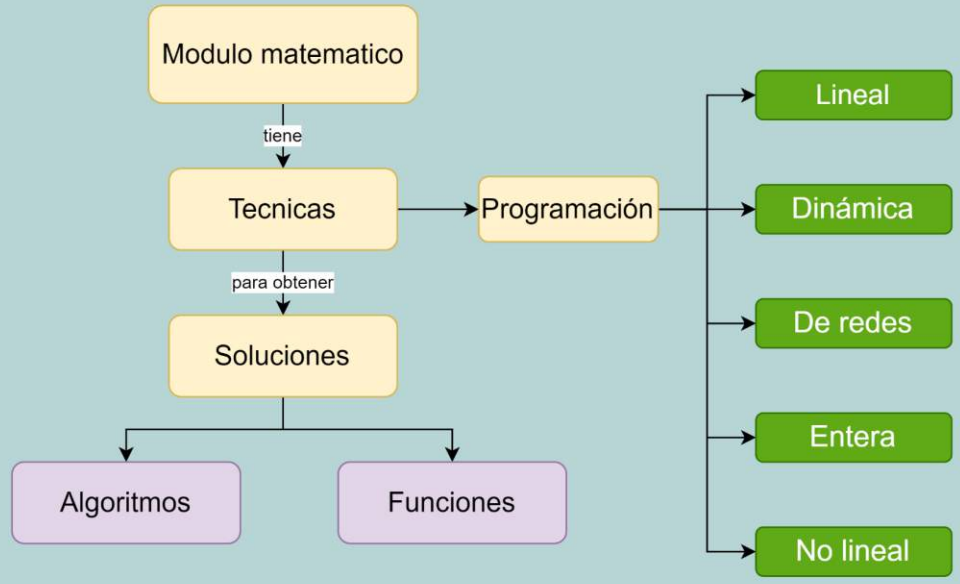
*En la investigación de operaciones no se tiene una sola **técnica general** con la que se resuelven todos los modelos matemáticos.*

*Más bien, la clase y la complejidad del modelo **determina** la naturaleza del método de solución*

Entre estas están programación..

1	**Lineal**	la tecnica más importante de investigación de operaciones, es diseñada para modelos con funciones y restricciones estrictamente **lineales**.
2	**Entera**	Es en la que las variables toman valores **enteros**.
3	**Dinámica**	en la que el modelo original se puede **descomponer** en subproblemas más pequeños
4	**De redes**	En la que el problema se puede modelar como una **red**
5	**No lineal**	en las que las funciones del modelo son **no lineales**

1.2 Solución del modelo de investigación de operaciones



*En general las soluciones no se obtienen en formas cerradas, es decir, parecidas a formulas, más bien, se determinan mediante **algoritmos**.*

Los Algoritmos

*Proporcionan reglas fijas de computo que se aplican en **forma repetitiva** al problema y cada **iteración**, obteniendo una solución cada vez más cerca a la optima.*

Los calculos

*Suelen ser **tediosos** es necesario ejecutarlos en una **computadora***

Si son muy complicados

*puede necesitarse abandonar una **solución optima** para buscar una **buena solución** usando Heurísticas o reglas simples.*

1.2 Solución del modelo de investigación de operaciones

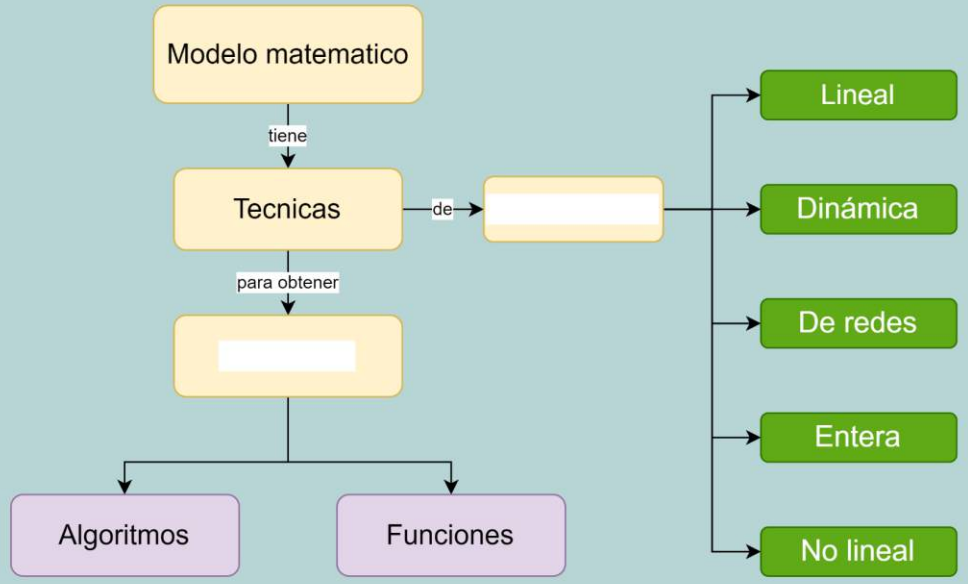
En la investigación de operaciones no se tiene una sola _____ *con la que se resuelven todos los modelos matemáticos.*

Más bien, la clase y la complejidad del modelo _____ *la naturaleza del método de solución*

Entre estas están programación..

1	_____	la tecnica más importante de investigación de operaciones, es diseñada para modelos con funciones y restricciones estrictamente _____.
2	_____	Es en la que las variables toman valores _____.
3	_____	en la que el modelo original se puede _____ en subproblemas más pequeños
4	_____	En la que el problema se puede modelar como una _____
5	_____	en las que las funciones del modelo son **no** _____

1.2 Solución del modelo de investigación de operaciones



En general las soluciones no se obtienen en formas cerradas, es decir, parecidas a formulas, más bien, se determinan mediante _____ .

Los Algoritmos

Proporcionan reglas fijas de computo que se aplican en _____ *al problema y cada* _____ *, obteniendo una solución cada vez más cerca a la optima.*

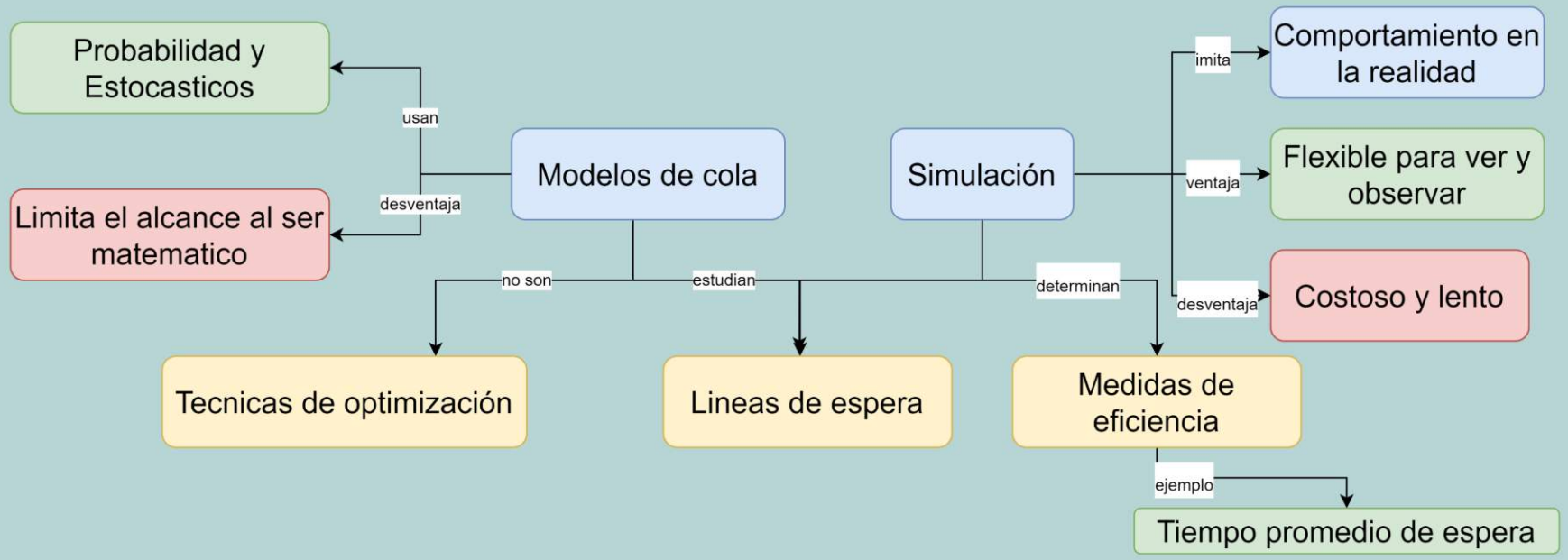
Los calculos

Suelen ser _____ *es necesario ejecutarlos en una* _____

Si son muy complicados

puede necesitarse abandonar una _____ *para buscar una* _____ *usando Heurísticas o reglas simples.*

1.3 Modelos de colas y simulación



Modelos de cola y simulación

tratan de estudiar las líneas de espera. No son tecnicas de optimización; más bien determinan medidas de deficiencia de las lineas de espera.

Los modelos de cola usan

Modelos de probabilidad y estocásticos para analizar las líneas de espera

y la simulación

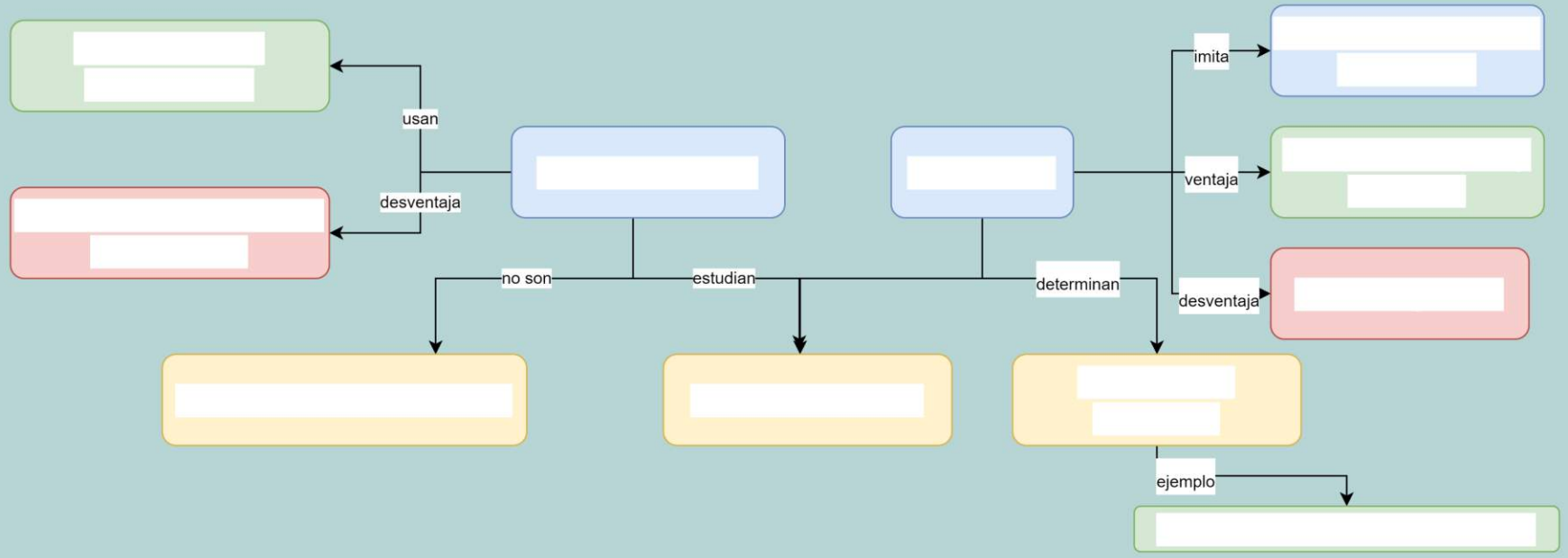
estima las medidas de eficiencia al imitar el comportamiento del sistema de la realidad.

Modelos de cola vs simulación

La simulación es casi lo mejor para observar un sistema real. La direrencia principal entre estas es que los modelos de colas sólo son matemáticos, y en consecuencia, están sujetos a hipótesis específicas que limitan el alcance de la aplicación. Por otro lado la simulación es flexible y con ella se puede analizar prácticamente cualquier caso de colas. Sin embargo la simulación es costosa en tiempo y recursos

1.3

Modelos de colas y simulación



Modelos de cola y simulación

tratan de estudiar las líneas de espera. No son **tecnicas de optimización**; más bien determinan **medidas de deficiencia** de las lineas de espera.

Los modelos de cola usan

Modelos de **probabilidad y estocásticos** para analizar las líneas de espera

y la simulación

estima las **medidas de eficiencia** al imitar el comportamiento del sistema de la realidad.

Modelos de cola vs simulación

La simulación es casi lo mejor para observar un **sistema real**. La direrencia principal entre estas es que los **modelos de colas sólo son matemáticos**, y en consecuencia, están sujetos a hipótesis específicas que limitan el alcance de la aplicación. Por otro lado la simulación es **flexible** y con ella se puede analizar prácticamente cualquier caso de colas. Sin embargo la simulación **es costosa en tiempo y recursos**

1.4 El Arte del modelado

La mayoría de la aplicaciones en IO requieren **diversas** fases de aproximación.

En primer paso

Se abstrae del **mundo real** el llamado **mundo real supuesto**.

Esto se logra

Concentrándonos en las **variables** dominantes que controlan el sistema del **mundo real** es decir concentrando varios parámetros del mundo real en un **parámetro único** del mundo real supuesto.

Figura 1.4

Ejemplo

La compañía Tyko manufacturing, produce piezas de plástico y contiene diversos departamentos como producción, materiales y ventas.
Esto hace que tengamos muchas variables para saber nuestro nivel de producción.
Pero podemos aproximarnos **concentrando** todo en 2 variables.
- Taza de producción.
- Taza de consumo.

1.5 Más que solo matemáticas

Tendemos a pensar que una solución de IO siempre está enraizado en el análisis **matemático**.

Sin embargo

Primero se deberían explorar métodos más **fáciles**.

Casos de estudio

UPS lanzo en 2004 el software ORION el cual resuelve el agente viajero para sus conductores. Pero los conductores resentían que una máquina fuera mejor que ellos, así que los desarrolladores pusieron un letrero a las hojas del itinerario que decía "**Gánale al computador**".
Los conductores se tomaron en serio el desafío y algunos encontraban nuevas rutas.

*Los viajeros en un aeropuerto se quejaban por la espera de su equipaje. Se tomó la decisión de colocar las puertas de llegada más lejos del área de recepción de equipaje, lo cual hacía que los pasajeros **caminaran más**, esto desapareció las quejas porque daba tiempo para que el equipaje se entregara el carrusel.*

En un edificio de oficinas que investigaba atender las quejas de espera del elevador, se decidió poner **espejos** de cuerpo completo a la entrada, las quejas desaparecieran.

Conclusión

*Un estudio de IO no debe iniciar con el pensamiento de utilizar una técnica matemática antes de que se **justifique** su uso.*
*Es decir cuando se utilice las técnicas más sencillas puede decidirse cuál era la **herramienta** más adecuada para la solución.*

Comportamiento humano

*Cualquier solución que no tome en cuenta el comportamiento **humano** quizás falle.*

1.4 El Arte del modelado

La mayoría de la aplicaciones en IO requieren _____ fases de aproximación.

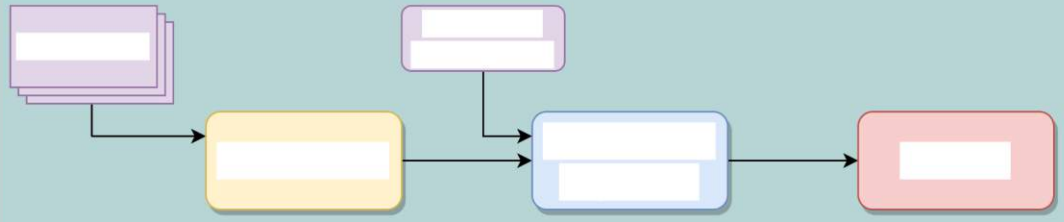
En primer paso

Se abstrae del _____ _____ el llamado _____ _____ _____.

Esto se logra

Concentrándonos en las _____ dominantes que controlan el sistema del _____ _____ es decir concentrando varios parámetros del mundo real en un _____ _____ del mundo real supuesto.

Figura 1.4



Ejemplo

La compañía Tyko manufacturing, produce piezas de plástico y contiene diversos departamentos como producción, materiales y ventas.
Esto hace que tengamos muchas variables para saber nuestro nivel de producción.
Pero podemos aproximarnos _____ todo en 2 variables.
- Taza de producción.
- Taza de consumo.

1.5 Más que solo matemáticas

Tendemos a pensar que una solución de IO siempre está enraizado en el análisis _____.

Sin embargo

Primero se deberían explorar métodos más _____.

Casos de estudio

UPS lanzo en 2004 el software ORION el cual resuelve el agente viajero para sus conductores. Pero los conductores resentían que una máquina fuera mejor que ellos, así que los desarrolladores pusieron un letrero a las hojas del itinerario que decía "_____".
Los conductores se tomaron en serio el desafío y algunos encontraban nuevas rutas.

Los viajeros en un aeropuerto se quejaban por la espera de su equipaje. Se tomó la decisión de colocar las puertas de llegada más lejos del área de recepción de equipaje, lo cual hacía que los pasajeros _____, esto desapareció las quejas porque daba tiempo para que el equipaje se entregara el carrusel.

En un edificio de oficinas que investigaba atender las quejas de espera del elevador, se decidió poner _____ de cuerpo completo a la entrada, las quejas desaparecieran.

Conclusión

Un estudio de IO no debe iniciar con el pensamiento de utilizar una técnica matemática antes de que se _____ su uso.
Es decir cuando se utilice las técnicas más sencillas puede decidirse cuál era la _____ más adecuada para la solución.

Comportamiento humano

Cualquier solución que no tome en cuenta el comportamiento _____ quizás falle.

2.1 Modelo de PL con dos variables

Analizaremos una solución gráfica de un problema de PL con 2 variables, este presenta fundamentos concretos para la siguiente sección

Ejemplo 2:1-1

La compañia Reddi mikks produce pinturas para interiores y exteriores con 2 materias primeas, M1 y M2.

Matería prima	Interior	Exterior	Disponiblidad diaria(Toneladas)
M1	6	4	24
M2	1	2	6
Uitlidad por tonelada($1000)	5	4	

A	La demanda diaria de pintura para interiores es máximo **2 toneladas**.
B	La demanda de exteriores siempre debe ser mayor a la de **interiores + 1**.

2.2 Componentes

Todos los modelos de IO constan de tres componentes.

A	**Variables de desición**
B	**El objetivo que necesitamos optimizar**
C	**Las restricciones que la solución debe satisfacer**

Variables

La definición de las variables es el primer paso esencial al desarrollo del modelo, para el problema es necesario determinar la cantidad diaria que se debe producir de pintura para la materia **interna** y **externa**.

x_1 = Toneladas producidas diariamente de pintura para exteriores

x_2 = Toneladas producidas diariamente de pintura para interiores

Objetivo

La meta de Reddy Mikks es maximizar la utilidad diaria total de ambas pinturas, por lo tanto tenemos:

Utilidad para exteriores = $5x_1$ miles de dólares

Utilidad para interiores = $4x_2$ miles de dólares

Entonces nuestro objetivo seria:

Maximizar $z = 5x_1 + 4x_2$

Restricciones

Las restricciones limitan la utilización de las materias primas y la demanda del producto, por lo tanto tenemos:

$$6x_1 + 4x_2 \leq 24 \text{ (M1)} \quad (1)$$

$$x_1 + 2x_2 \leq 6 \text{ (M2)} \quad (2)$$

Tenemos además que la producción diaria de pintura para interiores no debe exceder a la pintura para exteriores en más de 1 tonelada lo cual es:

$$x_2 + x_1 \leq 1 \text{ (Límite del mercado)} \quad (3)$$

Después tenemos que la demanda de pintura para interiores debe ser máximo 2 toneladas:

$$x_2 \leq 2 \text{ (Límite de demanda)} \quad (4)$$

Por ultimo tenemos una restricción implicita que requiere que todas las variables $x_1 y x_2$ asuman solo valores positivos o cero, estas son conocidas como restricciones de **no negatividad**.

$$x_1 \geq 0 \quad (5)$$

$$x_2 \geq 0 \quad (5)$$

2.1 Modelo de PL con dos variables

Analizaremos una solución gráfica de un problema de PL con 2 variables, este presenta fundamentos concretos para la siguiente sección

Ejemplo 2:1-1

La compañia Reddi mikks produce pinturas para interiores y exteriores con 2 materias primeas, M1 y M2.

Matería prima	Interior	Exterior	Disponiblidad diaria(Toneladas)
M1	6	4	24
M2	1	2	6
Uitlidad por tonelada($1000)	5	4	

A	La demanda diaria de pintura para interiores es máximo **2 toneladas**.
B	La demanda de exteriores siempre debe ser mayor a la de **interiores + 1**.

2.2 Componentes

Todos los modelos de IO constan de tres componentes.

A	
B	**El** **que necesitamos optimizar**
C	**Las** **que la solución debe satisfacer**

Variables

La definición de las variables es el primer paso esencial al desarrollo del modelo, para el problema es necesario determinar la cantidad diaria que se debe producir de pintura para la materia y .

x_1 = Toneladas producidas diariamente de pintura para exteriores

x_2 = Toneladas producidas diariamente de pintura para interiores

Objetivo

La meta de Reddy Mikks es maximizar la utilidad diaria total de ambas pinturas, por lo tanto tenemos:

Utilidad para exteriores = $5x_1$ miles de dólares

Utilidad para interiores = $4x_2$ miles de dólares

Entonces nuestro objetivo seria:

Maximizar $z =$

Restricciones

Las restricciones limitan la utilización de las materias primas y la demanda del producto, por lo tanto tenemos:

(M1) (1)

(M2) (2)

Tenemos además que la producción diaria de pintura para interiores no debe exceder a la pintura para exteriores en más de 1 tonelada lo cual es:

. (Límite del mercado) (3)

Después tenemos que la demanda de pintura para interiores debe ser máximo 2 toneladas:

(Límite de demanda) (4)

Por ultimo tenemos una restricción implicita que requiere que todas las variables $x_1 y x_2$ asuman solo valores positivos o cero, estas son conocidas como restricciones de .

(5)

(5)

El modelo completo es:

Maximizar $z = 5x_1 + 4x_2$	
$6x_1 + 4x_2 \leq 24$ (M1)	(1)
$x_1 + 2x_2 \leq 6$ (M2)	(2)
$x_2 + x_1 \leq 1$ (Límite del mercado)	(3)
$x_2 \leq 2$ (Límite de demanda)	(4)
$x_1 \geq 0$	(5)
$x_2 \geq 0$	(5)

Solución factible

Todos los valores de x_1 y x_2 que cumplen estas restricciones constituyen una **solución factible**.

Infinidad de soluciones

La meta de este problema es determinar la solución óptima.

Primero es necesario demostrar que este problema tiene un número **infinito** de soluciones factibles, esto es una propiedad que comparten todas las PL no triviales.

Esto significa que no podemos resolverlo por **enumeración**

Comentarios

La función objetivo y las restricciones de todas las PL deben ser **lineales**.

Se deben conocer con certeza todos los parámetros (**Coeficientes** de la función y las restricciones).

Solución gráfica de PL

Una solución rafica incluye 2 pasos:

1	**Determinar el espacio de soluciones factible**
2	**Determinar la solución óptima en ese espacio**

Ejemplo de Reddy Mikks

Paso 1) Determinar espacio de soluciones factibles.

Considera primero las restricciones de no negatividad $x_1 \geq 0$ y $x_2 \geq 0$. Estas restricciones limitan los valores de las variables al primer **cuadrante**.

Para las otras 4 restricciones debemos primero sustituir cada desigualdad con una ecuación.
Despues localizamos 2 puntos diferentes, sustituyendo cada variable por 0 y despejando la variable.

Ejemplo

$x_1 = 0$, obtenemos $x_2 = 6$
$x_2 = 0$, obtenemos $x_1 = 4$

Por último consideramos la dirección de la desigualdad, esta divide al plano en 2, y solo una de estas satisface la desigualdad.
Para esto seleccionamos el origen (0,0) como **punto de referencia** si este punto satisface la desigualdad entonces su lado es factible, de lo contrario el otro lado se convierte en factible.

Ejemplo

$6x_1 + 4x_2 \leq 24$ -> $6(0) + 4(0) \leq 24 = 0 \leq 24$

Como es menor que 24 el plano que contiene (0,0) es factible.

El modelo completo es:

Maximizar $z = 5x_1 + 4x_2$	
$6x_1 + 4x_2 \leq 24$ (M1)	(1)
$x_1 + 2x_2 \leq 6$ (M2)	(2)
$x_2 + x_1 \leq 1$ (Límite del mercado)	(3)
$x_2 \leq 2$ (Límite de demanda)	(4)
$x_1 \geq 0$	(5)
$x_2 \geq 0$	(5)

Solución factible

Todos los valores de x_1 y x_2 que cumplen estas restricciones constituyen una **solución** _____.

Infinidad de soluciones

La meta de este problema es determinar la solución óptima.

Primero es necesario demostrar que este problema tiene un número _____ de soluciones factibles, esto es una propiedad que comparten todas las PL no triviales.

Esto significa que no podemos resolverlo por _____

Comentarios

La función objetivo y las restricciones de todas las PL deben ser _____.

Se deben conocer con certeza todos los parámetros (_____ de la función y las restricciones).

Solución gráfica de PL

Una solución rafica incluye 2 pasos:

1	**Determinar el espacio de soluciones factible**
2	**Determinar la solución óptima en ese espacio**

Ejemplo de Reddy Mikks

Paso 1) _____.

Considera primero las restricciones de no negatividad $x_1 \geq 0$ y $x_2 \geq 0$. Estas restricciones limitan los valores de las variables al primer _____.

Para las otras 4 restricciones debemos primero sustituir cada desigualdad con una ecuación.
Despues localizamos 2 puntos diferentes, sustituyendo cada variable por 0 y despejando la variable.

Ejemplo [redacted]

$x_1 = 0$, obtenemos $x_2 = 6$
$x_2 = 0$, obtenemos $x_1 = 4$

Por último consideramos la dirección de la desigualdad, esta divide al plano en 2, y solo una de estas satisface la desigualdad.
Para esto seleccionamos el origen (0,0) como _____ si este punto satisface la desigualdad entonces su lado es factible, de lo contrario el otro lado se convierte en factible.

Ejemplo [redacted]

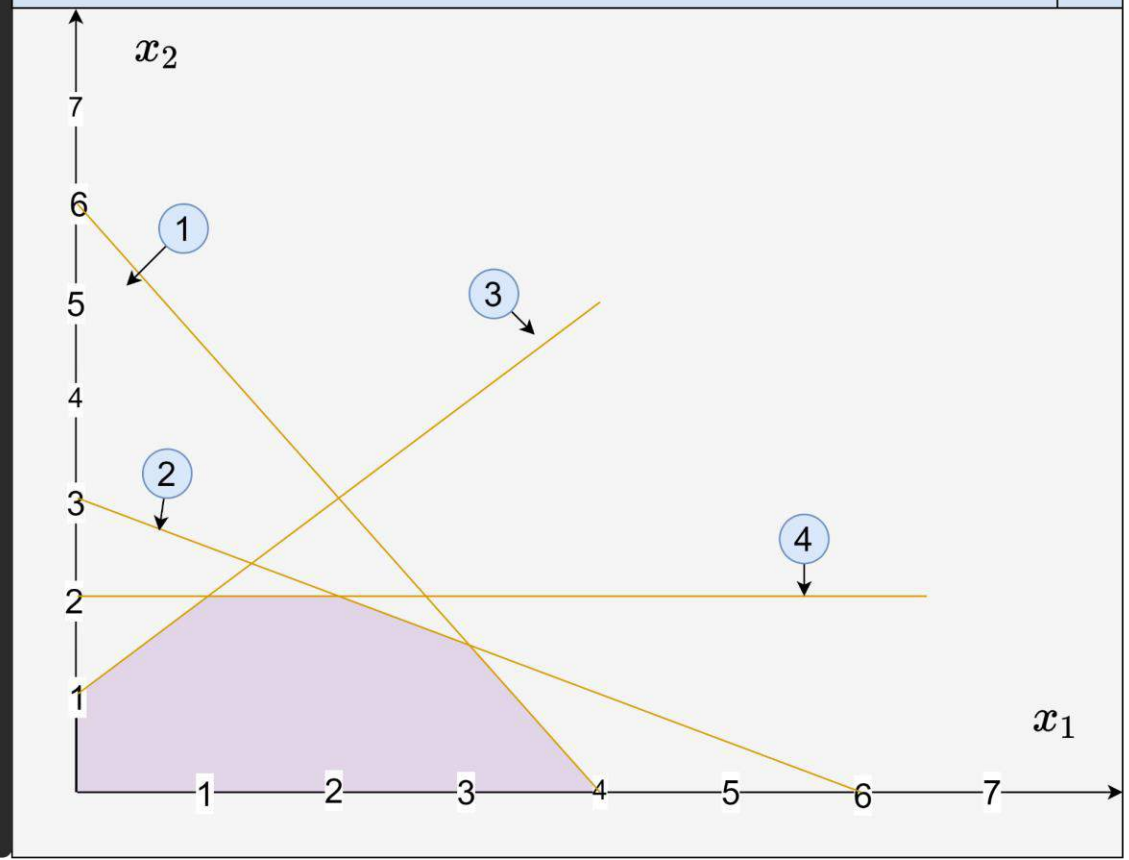
$6x_1 + 4x_2 \leq 24$	->	$6(0) + 4(0) \leq 24$ = $0 \leq 24$

Como es menor que 24 el plano que contiene (0,0) es factible.

EJERCICIO

Maximizar $z = 5x_1 + 4x_2$

$6x_1 + 4x_2 \leq 24$ (M1)	(1)
$x_1 + 2x_2 \leq 6$ (M2)	(2)
$x_2 + x_1 \leq 1$ (Límite del mercado)	(3)
$x_2 \leq 2$ (Límite de demanda)	(4)
$x_1 \geq 0$	(5)
$x_2 \geq 0$	(5)



1	Primera restricción	3	tercera restricción
$6x_1 + 4x_2 = 24$		$-x_1 + x_2 = 1$	
$6(0) + 4x_2 = 24$		$-(0) + x_2 = 1$	
$x_2 = \frac{24}{4} = 6$		$-x_1 = 1$	
$6x_1 + 4(0) = 24$		$x_1 = -1$	
$x_1 + \frac{24}{6} = 4$			
Dirección		Dirección	
$6(0) + 4(0) \leq 24$		$(0) + (0) \leq 1$	
$0 \leq 24$			

2	segunda restricción	4	cuarta restricción
$x_1 + 2x_2 \leq 6$		$x_2 \leq 2$	
$(0) + 2x_2 = 6$			
$x_2 = \frac{6}{2} = 3$			
$x_1 + (0) = 6$		$x_2 = 2$	
$x_1 = 6$			
Dirección		Dirección	
$(0) + 2(0) \leq 6$		$0 \leq 2$	
$(0) \leq 6$			

EJERCICIO

Maximizar $z = 5x_1 + 4x_2$

$$6x_1 + 4x_2 \leq 24 \text{ (M1)} \quad (1)$$

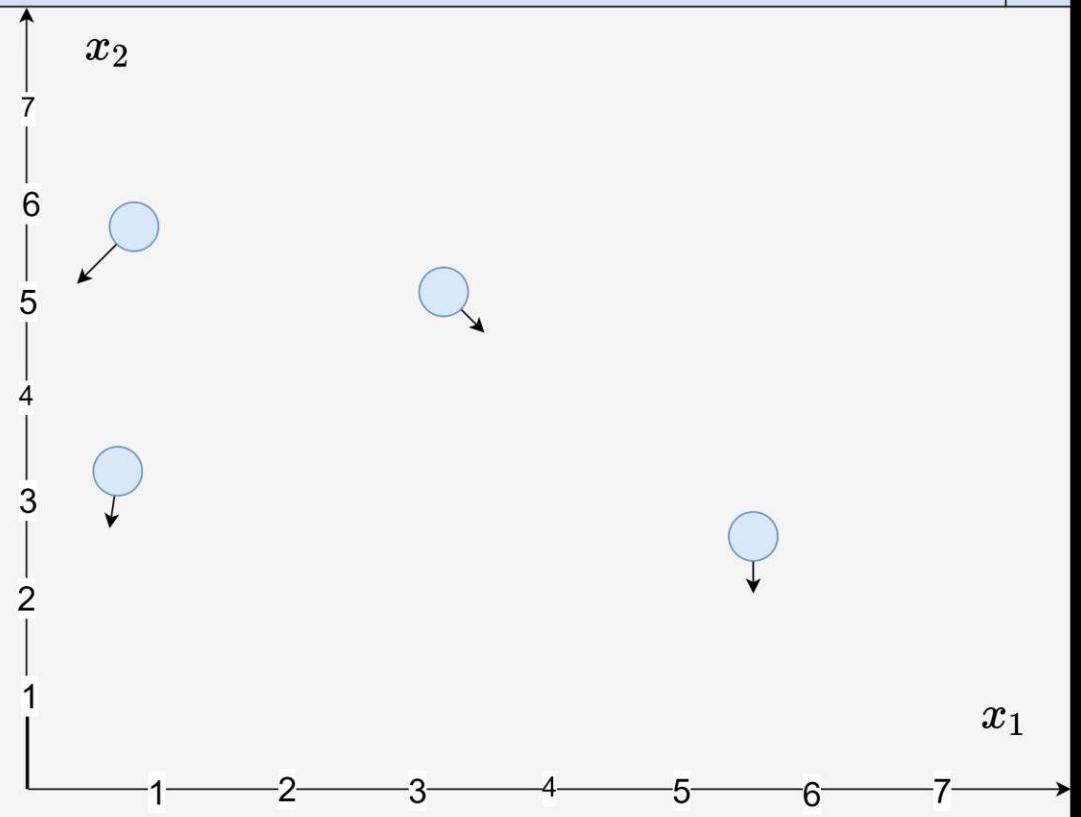
$$x_1 + 2x_2 \leq 6 \text{ (M2)} \quad (2)$$

$$x_2 + x_1 \leq 1 \text{ (Límite del mercado)} \quad (3)$$

$$x_2 \leq 2 \text{ (Límite de demanda)} \quad (4)$$

$$x_1 \geq 0 \quad (5)$$

$$x_2 \geq 0 \quad (5)$$



1	Primera restricción	3	tercera restricción
Dirección		Dirección	
2	segunda restricción	4	cuarta restricción
Dirección		Dirección	

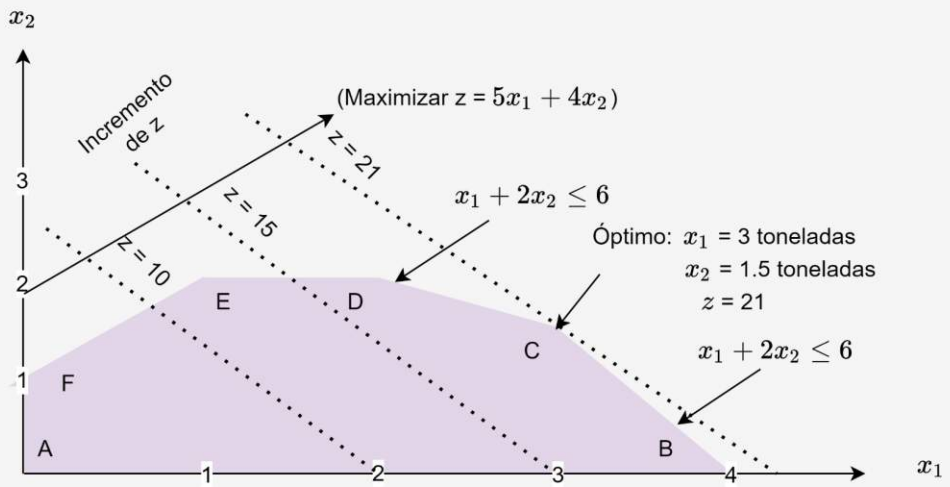
Paso 2) Determinación de la solución óptima

La cantidad de puntos de solución en el espacio factible es infinita, para resolverlo se requiere un procedimiento sistematico.

A) La meta de este problema es determinar la solución óptima.

Por ejemplo

Revisa la figura te daras cuenta que la solución tiene una dirección a medida que z aumenta



Al movernos en esta direccíón la salución optima ocurren en C.

Los valores asociadas de x_1 y x_2 con el punto óptima C se deeterminan resolviendo las ecuaciones asociadas a las lineas (1) y (2)

$$6x_1 + 4x_2 = 24$$

$$x_1 + 2x_2 = 6$$

La solución es $x_1 = 3$ y $x_2 = 1.5$ con $z = 21$, por lo tanto la utilidad diaria es de $21,000.

Paso 2) Determinación de la solución óptima

Comentarios

En la practica una PL puede tener cientos o miles de variables y restricciones, entonces ¿de que sirve una PL con 2 variables?

La respuesta es que la solución gráfica proporciona algo clave.

La solución de un PL, siempre esta en uno de los puntos de esquina del espacio de solución.

Esto nos limita la búsqueda del óptimo de un número infinito de soluciones factibles a un número finito de puntos de esquina.

Esto es la base del método simplex.

Solución PL con Excel Solver

En la practica los modelos de PL suelen tener miles de variables y la computadora es el único medio viable para resolverlos.

La hoja de calculo de excel es el medio de entrada y salida para la PL

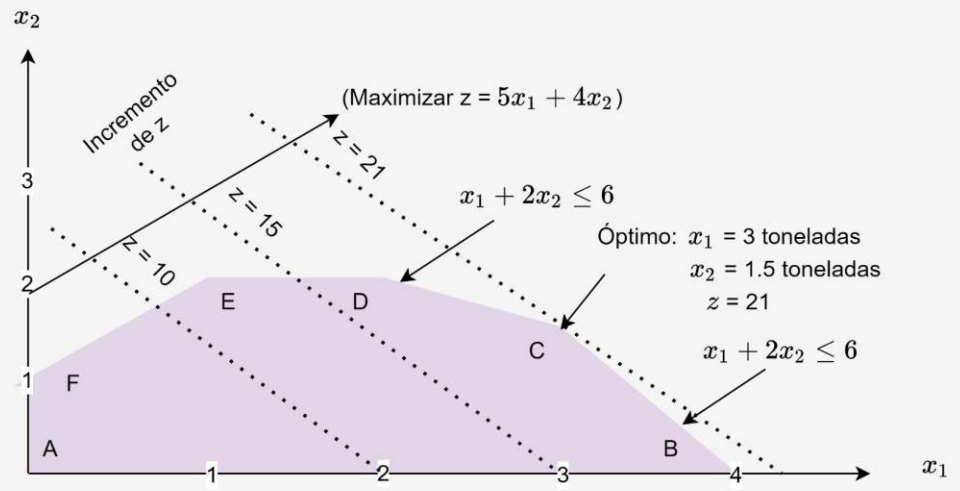
Paso 2) Determinación de la solución óptima

La cantidad de puntos de solución en el espacio factible es infinita, para resolverlo se requiere un procedimiento sistematico.

A) La meta de este problema es determinar la solución óptima.

Por ejemplo

Revisa la figura te daras cuenta que la solución tiene una dirección a medida que z aumenta



Al movernos en esta direccíón la salución optima ocurren en .

Los valores asociadas de x_1 y x_2 con el punto óptima C se deeterminan resolviendo las ecuaciones asociadas a las lineas (1) y (2)

$$6x_1 + 4x_2 = 24$$

$$x_1 + 2x_2 = 6$$

La solución es $x_1 = 3$ y $x_2 = 1.5$ con $z = 21$, por lo tanto la utilidad diaria es de $21,000.

Paso 2) Determinación de la solución óptima

Comentarios

En la practica una PL puede tener cientos o miles de variables y restricciones, entonces ¿de que sirve una PL con 2 variables?

La respuesta es que la solución gráfica proporciona algo clave.

La solución de un PL, siempre esta en uno de los puntos de esquina del espacio de solución.

Esto nos limita la búsqueda del óptimo de un número infinito de soluciones factibles a un número finito de puntos de esquina.

Esto es la base del método .

Solución PL con Excel Solver

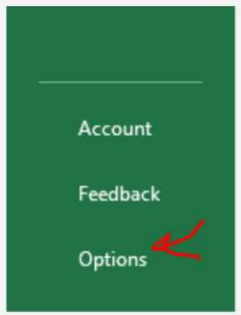
En la practica los modelos de PL suelen tener miles de variables y la computadora es el único medio viable para resolverlos.

La hoja de calculo de excel es el medio de entrada y salida para la PL

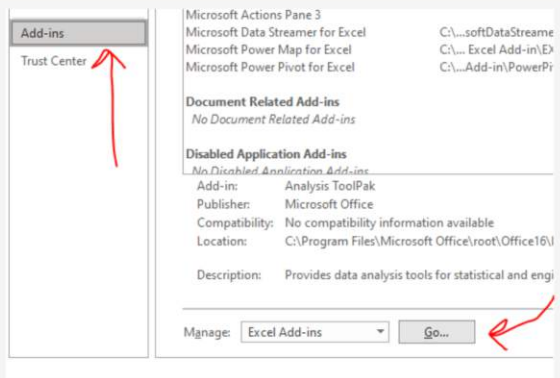
Solución PL con Excel Solver

Instalación

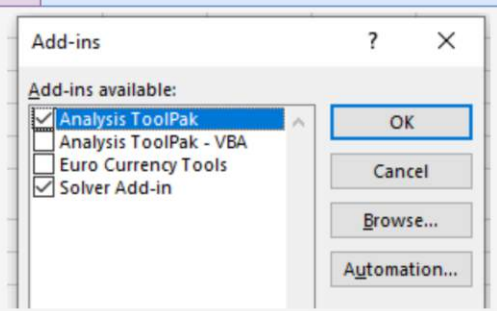
1 Primero accesamos al apartado de opciones



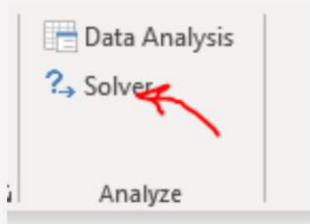
2 Entramos a Add-ins



3 Habilitamos los add-ins



4 Accesamos en "Data"



Configuración

Pensando en el problema de Reddy Mikks (Archivo SolverRM1.xls)

(1)	Celdas para ingresar datos (B5:C9) y (F6:F9)
(2)	Celdas que representan variables y la función objetivo (B13:D13)
(3)	Definiciones algebraicas de la función objetivo y el lado izquierda de las restricciones (D5:D9)
(4)	Celdas que proporcionan nombres y símbolos explicativos

El siguiente orden es el más conveniente y se recomienda su uso

	A	B	C	D	E	F
1		Reddy Mikks Model				
2	**Input data:**					
3		**x1**	**x2**			
4		**Exterior**	**Interior**	**Totales**		**Limites**
5	**Objetivo:**	5	4	0		
6	**Materia prima 1:**	6	4	0	<=	24
7	**Materia prima 2:**	1	2	0	<=	6
8	**Limite de producción:**	-1	1	0	<=	1
9	**Limite de demanda:**	0	1	0	<=	2
10		**>= 0**	**>= 0**			
11	**Output results:**					
12		x1	x2	z		
13	Solución					

Configuración

Para vincular solver con los datos de la hoja, proporcionaremos definiciones algebraicas de la función objetivo y restricciones.

Esto lo hacemos en los datos de entrada (B5:C9 y F6:F9)

A continuación colocamos las siguientes formulas:

	Expresión algebraica	Fórmula en la hoja de cálculo	Ingresa en la celda
Objetivo, z	$5x_1 + 4x_2$	=B5*$B$13+C5*$C$13	D5
Restricción 1	$6x_1 + 4x_2$	=B6*$B$13+C6*$C$13	D6
Restricción 2	$x_1 + 2x_2$	=B7*$B$13+C7*$C$13	D7
Restricción 3	$-x_1 + x_2$	=B8*$B$13+C8*$C$13	D8
Restricción 4	$0x_1 + x_2$	=B9*$B$13+C9*$C$13	D9

Sin embargo esto no es practico en PL grandes, en su lugar, la formula en la celda D5 se escribe en forma compacta como sigue:

=SUMPRODUCT(B5:C5,$B$13:$C$13)

Ahora todos nuestros elementos de PL estan listos para ejecutar el modelo, hagamos click en el menú solver de la barra de herramientas de la hoja de calculo.

y a continuación actualize el cuadro de diálogo como sigue:

Set Target cell	$D$5
Equal to	MAX
By changing cell	$B$13:$C$13

Esto indica al solver que las celdas B13 y C13 se utilizan para maximizar la función objetivo en la celda $D$5

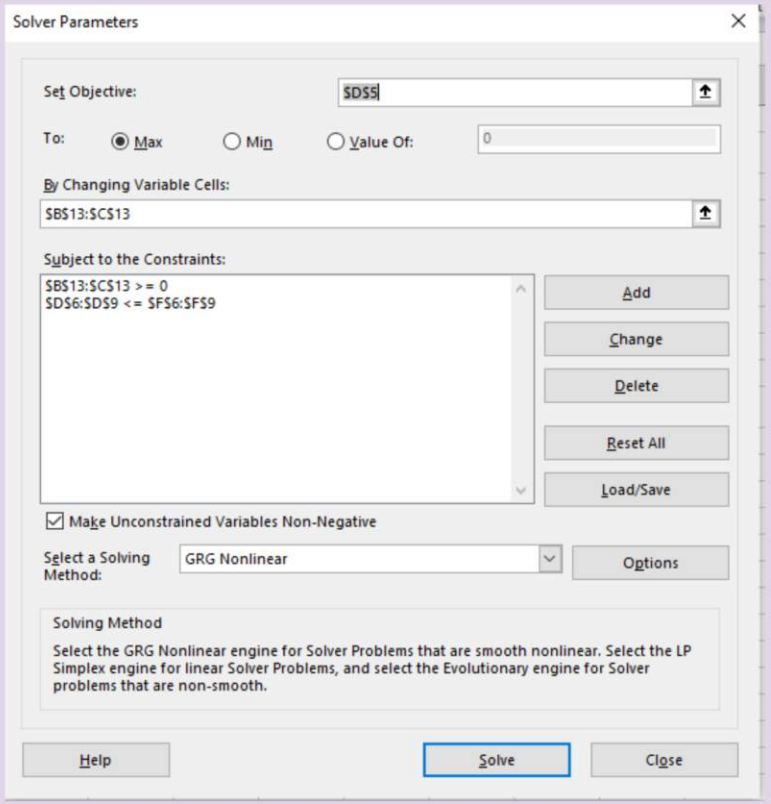
Configuración

Para establecer restricciones, haga click en el botón add para desplegar el cuadro add constraint y ponga:

$$\$D\$6 : \$D\$9 <= \$F\$6 : \$F\$9$$

Para las de no negatividad añada una vez mas

$$\$B\$13 : \$C\$13 >= 0$$



Al dar Solve resolveremos el problema de PL.

Problemas

2 Problema de asignación

*se tienen n tareas y n máquinas. Existe un **costo** cij por realizar la tarea i con máquina j.*

se desea

***minimizar** el coste total de operación de modo que*

1	cada tarea se asigna a una y sólo una máquina.
2	cada máquina realice una y sólo una tarea.

2.1 Modelo matemático

Variables

1	**Asignado**	x_{ij}

1 si la tarea ***i*** se hace con la máquina ***j***, ***0*** en caso contrario.

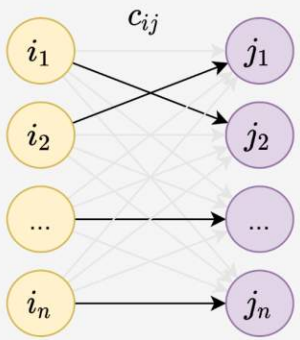
Parametros

1	**Número de tareas y máquinas**	**n**
2	**costo de realizar la tarea i con la maquina j**	c_{ij}

Función objetivo

Mínimizar

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$



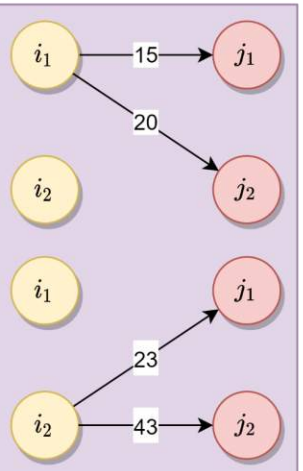
Restricciones

1	$\sum_{j=1}^{n} x_{ij} = 1$	$\forall i, i = 1, .., n.$
2	$\sum_{i=1}^{n} x_{ij} = 1$	$\forall j, j = 1, .., n$
3	$x_{ij} \in \{0, 1\}$	$\forall i, j \quad i, j \in 1, .., n$

2 Problema de asignación

Ejercicio

Id	j_1	j_2
i_1	15	20
i_2	23	43

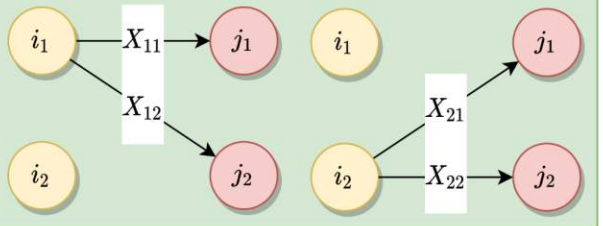


Función objetivo

$$min \quad (15(X_{11}) + 20(X_{12})) + (23(X_{21}) + 43(X_{22}))$$

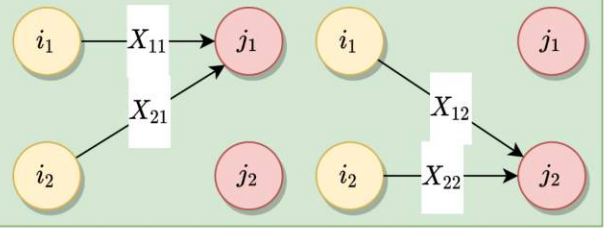
Restriccion 1

$X_{11} + X_{12} = 1$	$i = 1$
$X_{21} + X_{22} = 1$	$i = 2$



Restriccion 2

$X_{11} + X_{21} = 1$	$j = 1$
$X_{12} + X_{22} = 1$	$j = 2$



2 Problema de asignación

se tienen n tareas y n máquinas. Existe un *cij por realizar la tarea i con máquina j.*

se desea

el coste total de operación de modo que

1	cada tarea se asigna a una y sólo una máquina.
2	cada máquina realice una y sólo una tarea.

2.1 Modelo matemático

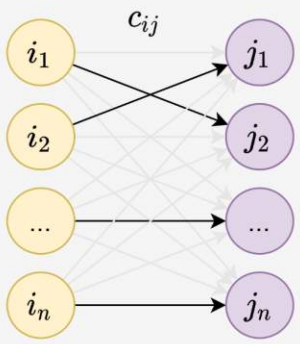
Variables

1		x_{ij}

1 si la tarea ***i*** se hace con la máquina ***j***, ***0*** en caso contrario.

Parametros

1		**n**
2		c_{ij}



Función objetivo

Mínimizar	

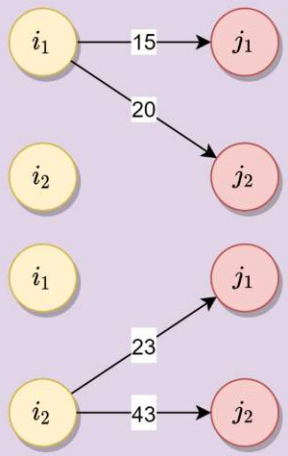
Restricciones

1		$\forall i, i = 1, .., n.$
2		$\forall j, j = 1, .., n$
3	$x_{ij} \in \{0, 1\}$	$\forall i, j \quad i, j \in 1, .., n$

2 Problema de asignación

Ejercicio

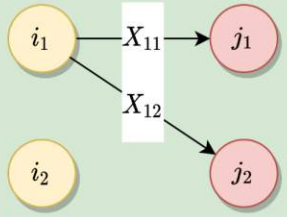
Id	j_1	j_2
i_1	15	20
i_2	23	43

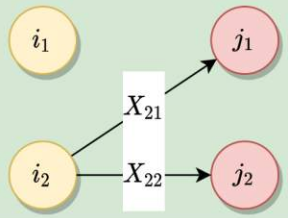


Función objetivo

Restriccion 1

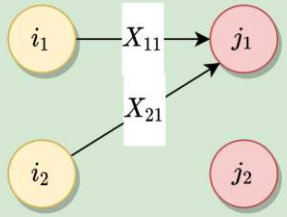
	$i = 1$
	$i = 2$

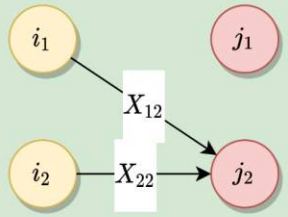




Restriccion 2

	$j = 1$
	$j = 2$





1 Knaspack problem

*Tambien llamado problema de la **mochila**.*

Busca

Seleccionar los objetos que vamos a llevar en la mochila.

Sin exceder

La capacidad de la misma de modo que el **beneficio** que aporta estos objetos sea el mayor posible.

1.1 Modelo matemático

Variables

1	**Seleccionado**	x_i

1 si el articulo ***i*** es escogido.
0 si el articulo ***i*** no es escogido.

Parametros

1	**beneficio de artículo**	v_i
2	**peso de artículo**	w_i
3	**Capacidad de mochila**	Q

Función objetivo

Maximizar $$\sum_{i=1}^{n} v_i x_i$$

Restricciones

$$\sum_{i=1}^{n} w_i x_i \leq Q \qquad x_i \in \{0,1\}, i = 1,..,n$$

2 Knaspack problem

Ejercicio

Id	X	w	v
1	X_1	1	1
2	X_2	2	6
3	X_3	5	18
4	X_4	6	22

Función objetivo

$$max \quad (1)X_1 + (6)X_2 + (18)X_3 + (22)X_4$$

Restriccion

$$(1)X_1 + (6)X_2 + (18)X_3 + (22)X_4$$

1 Knaspack problem

Tambien llamado problema de la .

Busca

los objetos que vamos a llevar en la mochila.

Sin exceder

La capacidad de la misma de modo que el que aporta estos objetos sea el mayor posible.

1.1 Modelo matemático

Variables

1		x_i

1 si el articulo ***i*** es escogido.
0 si el articulo ***i*** no es escogido.

Parametros

1		v_i
2		w_i
3		Q

Función objetivo

Restricciones

$$x_i \in \{0,1\}, i = 1, .., n$$

2 Knaspack problem

Ejercicio

Id	X	w	v
1	X_1	1	1
2	X_2	2	6
3	X_3	5	18
4	X_4	6	22

Función objetivo

Restriccion

3 Problema de cobertura de conjuntos

Tenemos un cierto **m** número de regiones, y además tenemos un **n** número de centros de servicio.

Centros de servicio

Para cada centro de servicio conocemos su costo de instalación y además conocemos las regiones que cubre.

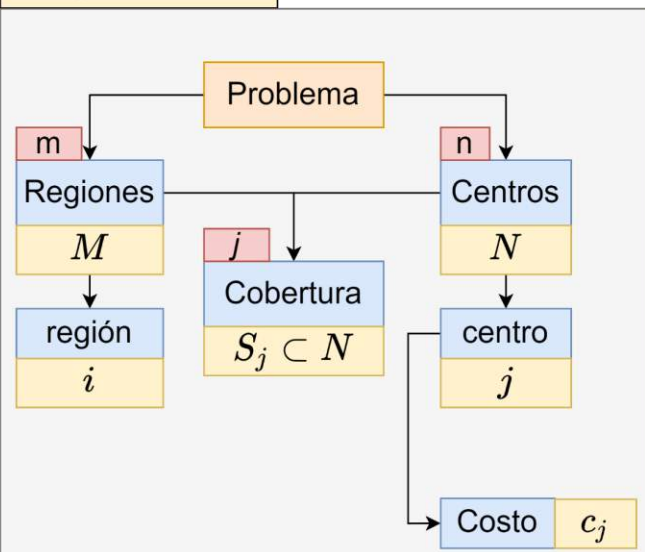
Objetivo

Deseamos escoger un conjunto de centros de servicio de tal manera que se cubran todas las regiones y su costo de instalación sea el minimo.

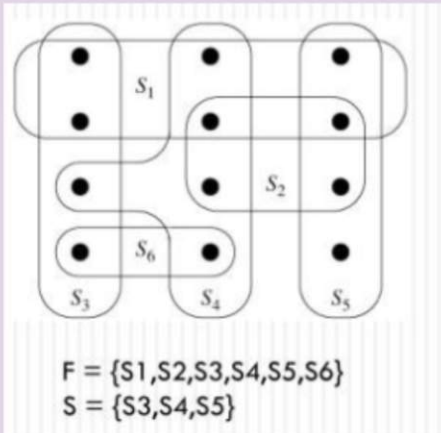
Matriz de adyacencia

Tenemos una matriz de adyacencia **A**, el cual nos indica si una región es cubierta por un centro de servicio.

Figura 3.1



Ejemplo de problema de cobertura



1.1 Modelo matemático

Variables			
1	Si centro seleccionado	x_j	***1*** si el centro j es seleccionado. ***0*** si no.
2	Costo de instalación de j	c_j	

Parametros		
1	Si i esta en el campo de cobertura de j	a_{ij}
2	Conjunto de centros i	N
3	Conjunto de regiones j	M
4	Campo de cobertura de j	S_j
5	Número de centros	***n***
6	Número de regiones	***m***

Función objetivo

Minimizar	$\sum_{j=1}^{n} c_j x_j.$

Restricciones

1	$x_j \in \{0,1\} \quad for \quad j = 1, \ldots, n.$	*Restriccion de variable entera.*
2	$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad for \quad i, \ldots, m.$	*Restriccion de todas las regiones deben ser cubiertas.*

3 Problema de cobertura de conjuntos

Tenemos un cierto **m** número de _____, y además tenemos un **n** número de _____.

Centros de servicio

Para cada centro de servicio conocemos su _____ de instalación y además conocemos las regiones que cubre.

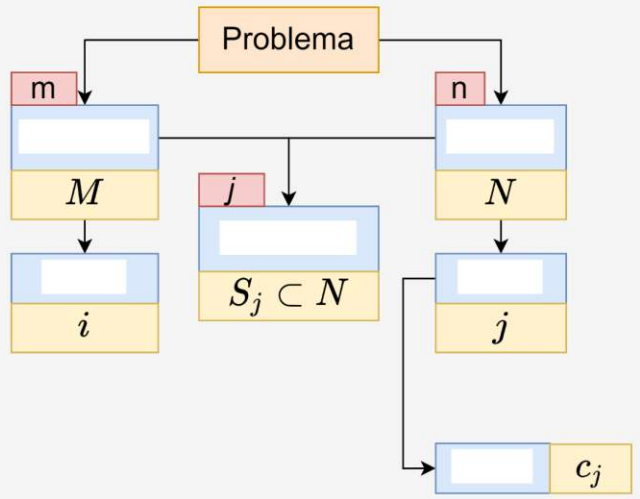
Objetivo

Deseamos escoger un conjunto de centros de servicio de tal manera que se cubran todas las regiones y su costo de instalación sea el _____.

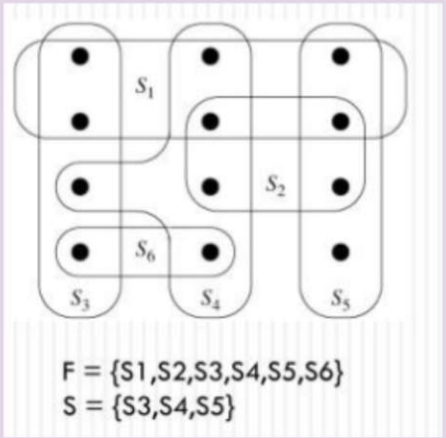
Matriz de adyacencia

Tenemos una matriz de adyacencia **A**, el cual nos indica si una región es cubierta por un centro de servicio.

Figura 3.1



Ejemplo de problema de cobertura



1.1 Modelo matemático

Variables

1	Si centro seleccionado		***1*** si el centro j es seleccionado. ***0*** si no.
2	Costo de instalación de j		

Parametros

1	Si i esta en el campo de cobertura de j	
2	Conjunto de centros i	
3	Conjunto de regiones j	
4	Campo de cobertura de j	
5	Número de centros	
6	Número de regiones	

Función objetivo

Minimizar	

Restricciones

1		*Restriccion de variable entera.*
2		*Restriccion de todas las regiones deben ser cubiertas.*

MLP

1 MLP o problema de latencia mínima

Introducción

Es un problema importante en el area de la manufacturación y **asignación** de maquinas.

Fue introducido

En **1973** por Fox

Es conocido como

El problema del repartidor o el problema del reparador

En ingles

Delivery man problem or traveling repairman problem(**TRP**)

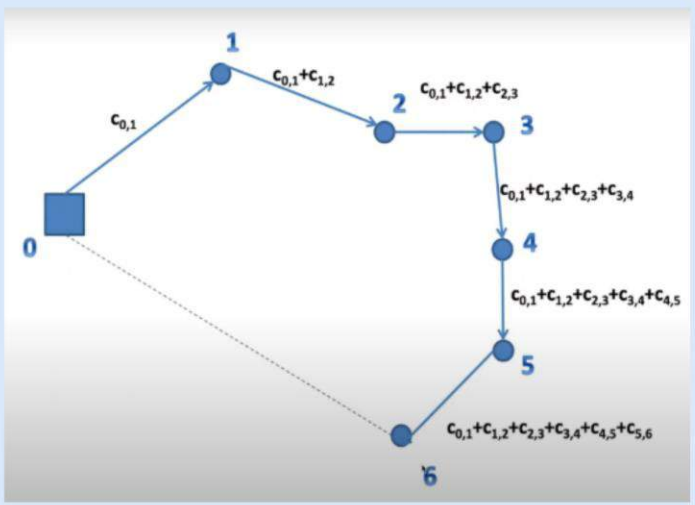
Su complejidad es

NP-Hard

MLP

Este problema consiste en encontrar la permutación con el menor valor de **latencia** L.

Figura 1.1



2 MLP Modelo

Grafo

Dado un grafo completo $G = (V, A)$ donde $V = \{0, 1, 2, \ldots, n\}$ es el conjunto de nodos y A es el conjunto de arcos.
Nodo 0 correspondo al nodo **raíz**.

Matriz de costos

Sea $C = (c_{ij})$ una matriz de costos no negativos asociados a los arcos $(i, j) \in A$

Permutación

Dada una permutación(**ruta hamiltoniana**) de nodos $P = \{0, [1], [2], [3], \ldots, [n]\}$ de V con el nodo 0 en la primera posición.

Latencia

La latencia $l_{[i]}$ del nodo de la posición i es definida como la longitud de la ruta del nodo 0 al nodo [i] y es calculada como:

$$l[i] = \sum_{j=1}^{i} c_{[j-1][j]}$$

donde [i] es el índice del nodo de la posición i de la ruta y [0] = 0.

Latencia total

La latencia total de una permutacion P es la sumatoria de todas las latencias de los nodos y puede ser calculada como:

$$L = \sum_{i=1}^{n} l_{[i]} = \sum_{i=1}^{n} (n - i + 1) c_{[i-1][i]}$$

1 MLP o problema de latencia mínima

Introducción

Es un problema importante en el area de la manufacturación y de maquinas.

Fue introducido

En por Fox

Es conocido como

El problema del repartidor o el problema del reparador

En ingles

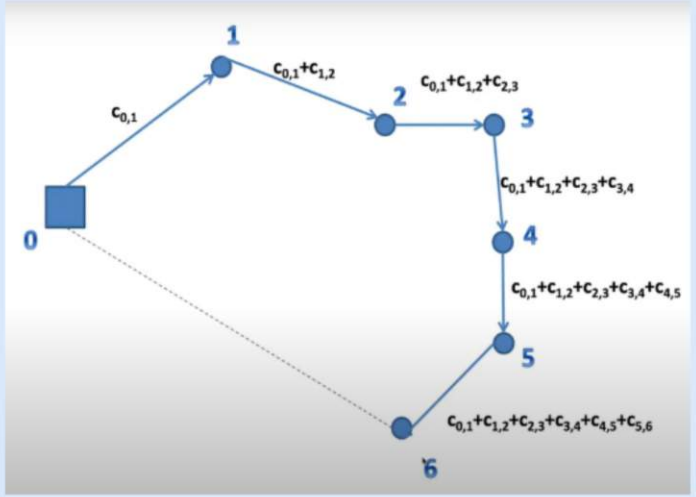
Delivery man problem or traveling repairman problem( )

Su complejidad es

MLP

Este problema consiste en encontrar la permutación con el menor valor de L.

Figura 1.1



2 MLP Modelo

Grafo

Dado un grafo completo $G = (V, A)$ donde $V = \{0, 1, 2, \ldots, n\}$ es el conjunto de nodos y A es el conjunto de arcos.
Nodo 0 correspondo al nodo .

Matriz de costos

Sea $C = (c_{ij})$ una matriz de costos no negativos asociados a los arcos $(i, j) \in A$

Permutación

Dada una permutación( ) de nodos $P = \{0, [1], [2], [3], \ldots, [n]\}$ de V con el nodo 0 en la primera posición.

Latencia

La latencia $l_{[i]}$ del nodo de la posición i es definida como la longitud de la ruta del nodo 0 al nodo [i] y es calculada como:

$$l[i] =$$

donde [i] es el índice del nodo de la posición i de la ruta y [0] = 0.

Latencia total

La latencia total de una permutacion P es la sumatoria de todas las latencias de los nodos y puede ser calculada como:

$$L = \sum_{i=1}^{n} l_{[i]} =$$

2 Problema de latencia mínima MLP

*Tambien llamado problema de la **reparador o repartidor**.*

Tenemos ***n*** clientes y conocemos el tiempo de servicio de cada cliente ***i***, S_i, conocemos la ubicacíón de los clientes por lo tanto conocemos las distancias a cada uno y su tiempo de traslado t_{ij}.

Matriz de Costos

Con eso tenemos la matriz de costos calculada como

$$c_{ij} = s_i + t_{ij}$$

Buscamos

Seleccionar la ruta que minimize el tiempo de espera total de los clientes.

De manera que pasemos por cada cliente una y solo una vez en la ruta.

1.1 Modelo matemático

Variables

1	**Ruta de i a j**	x_{ij}	***1*** si el agente toma la ruta de i a j. ***0*** si no.
2	**Latencia del nodo i**	l_i	

Parametros

1	**Tiempo de servicio del cliente i**	s_i
2	**Tiempo de traslado de i a j**	t_{ij}
3	**Costo de cliente i a j**	c_{ij}

Función objetivo

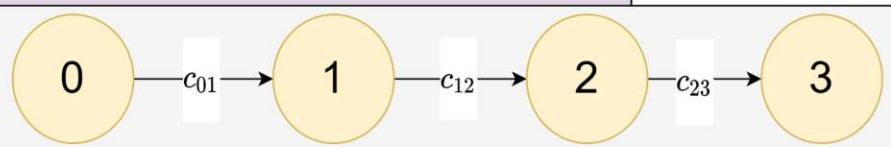
Minimizar $$\sum_{i=1}^{n} l_{[i]} = \sum_{i=1}^{n} (n-i+1) c_{[i-1][i]}$$

2 Problema de latencia mínima MLP

Restricciones

1	$\sum_{i=0}^{n} x_{ij} = 1$	$j = 1, 2, .., n$	*Restricciones para asegurar que un nodo tenga solamente una salida y una entrada.*
2	$\sum_{j=0}^{n} x_{ij} = 1$	$i = 1, 2, .., n$	
3	$x_{ij} \in \{0, 1\}$	$i, j = 1, 2, .., n$	*Restriccion de variable entera.*
4	$l_i \geq 0$	*Restriccion de variable entera.*	
5	$l_i - u_j + (T + c_{ij}) x_{ij}$	*Eliminación de subtours.*	
6	$c_{0i} x_{0i} \leq l_i$	*Latencia del nodo i sea mayor o igual a la latencia del nodo siguiente.*	

Ejercicio



¿Cual es el tiempo total que esperara el cliente 3?

$$(s_0 + t_{01}) + (s_1 + t_{12}) + (s_2 + t_{23})$$

2 Problema de latencia mínima MLP

Tambien llamado problema de la _____ ***o*** _____.

Tenemos ***n*** clientes y conocemos el tiempo de servicio de cada cliente ***i***, S_i, conocemos la ubicacíón de los clientes por lo tanto conocemos las distancias a cada uno y su tiempo de traslado t_{ij}.

Matriz de Costos

Con eso tenemos la matriz de costos calculada como

$$c_{ij} = \square$$

Buscamos

Seleccionar la ruta que minimize el tiempo de espera total de los clientes.

De manera que pasemos por cada cliente una y solo una vez en la ruta.

1.1 Modelo matemático

Variables

1	**Ruta de i a j**	$\square$	***1*** si el agente toma la ruta de i a j. ***0*** si no.
2	**Latencia del nodo i**	$\square$	

Parametros

1	**Tiempo de servicio del cliente i**	$\square$
2	**Tiempo de traslado de i a j**	$\square$
3	**Costo de cliente i a j**	$\square$

Función objetivo

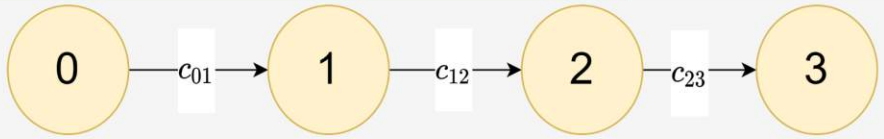
Minimizar $$\sum_{i=1}^{n} l_{[i]} = \sum_{i=1}^{n} (\square) c_{[i-1][i]}$$

2 Problema de latencia mínima MLP

Restricciones

1	$\sum_{i=0}^{n} \square = 1$	$j = 1, 2, .., n$	*Restricciones para asegurar que un nodo tenga solamente una salida y una entrada.*
2	$\sum_{j=0}^{n} \square = 1$	$i = 1, 2, .., n$	
3	$\square \in \{0, 1\}$	$i, j = 1, 2, .., n$	*Restriccion de variable entera.*
4	$l_i \geq 0$	*Restriccion de variable entera.*	
5	$\square \leq T$	*Eliminación de subtours.*	
6	$\square \leq l_i$	*Latencia del nodo i sea mayor o igual a la latencia del nodo siguiente.*	

Ejercicio



¿Cual es el tiempo total que esperara el cliente 3?

Ant colony

1 De hormigas reales a artificales

Introducción

Una colonia de hormigas es un sistema distribuido que a pesar de lo simple de sus individuos, juntos pueden realizar tareas complejas que exceden la capacidad de una unica **hormiga**.

La idea principal

Es que los principios de una colonia de hormigas puede ser usado como inspiración para una colonia **artificial** de agentes que colaboran para resolver problemas.

Esto se logra

Gracias a **stigmergy**, la cual es una forma de comunicación indirecta surgida de las hormigas.

Consiste

En un **químico** depositado en el suelo, el cual incrementa la probabilidad de que otras hormigas sigan el mismo camino.

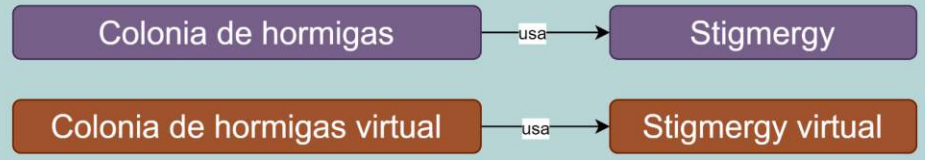
Algoritmos colonia de hormigas

Son algoritmos que usan una forma de stigmergy artificial para coordinarse

Por ejemplo

ACO o Ant colony optimization, El cual se enfoca en problemas de optimización discreta.

Figura 1



1 De hormigas reales a artificales

1.1 El comportamiento de la busqueda de comida de las hormigas

La facultad visual de las hormigas es muy **pobre**.

por eso

Su comunicación esta basada en el uso de quimicos producidos por ellas.

llamados Feromonas

particularmente algo llamado el **trail pheromone** o sendero de feromonas

Trail pheromone

Es un tipo especial de feromonas que las hormigas utilizan para señalar caminos en el suelo, este es la fuente de inspiración de ACO.

Figura 2

Hormigas dejando un rastro de feromonas.

1 De hormigas reales a artificales

Introducción

Una colonia de hormigas es un sistema distribuido que a pesar de lo simple de sus individuos, juntos pueden realizar tareas complejas que exceden la capacidad de una unica _____.

La idea principal

Es que los principios de una colonia de hormigas puede ser usado como inspiración para una colonia _____ de agentes que colaboran para resolver problemas.

Esto se logra

Gracias a _____, la cual es una forma de comunicación indirecta surgida de las hormigas.

Consiste

En un _____ depositado en el suelo, el cual incrementa la probabilidad de que otras hormigas sigan el mismo camino.

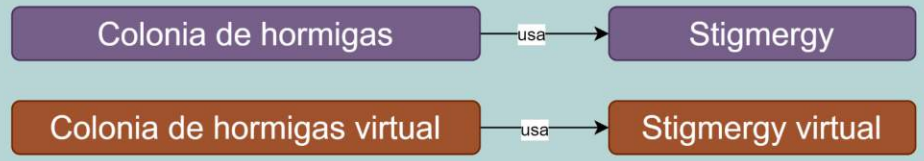
Algoritmos colonia de hormigas

Son algoritmos que usan una forma de stigmergy artificial para coordinarse

Por ejemplo

_____ o Ant colony optimization, El cual se enfoca en problemas de optimización discreta.

Figura 1



1 De hormigas reales a artificales

1.1 El comportamiento de la busqueda de comida de las hormigas

La facultad visual de las hormigas es muy _____.

por eso

Su comunicación esta basada en el uso de quimicos producidos por ellas.

llamados Feromonas

particularmente algo llamado el _____ _____ o sendero de feromonas

Trail pheromone

Es un tipo especial de feromonas que las hormigas utilizan para señalar caminos en el suelo, este es la fuente de inspiración de ACO.

Figura 2

Hormigas dejando un rastro de feromonas.

1 De hormigas reales a artificales

1.2 Experimentos de doble puente

El comportamiento de busqueda de comida de las hormigas ha sido investigado en experimentos por varios investigadores.

Por ejemplo

Deneubourg en 1990.

Consistio

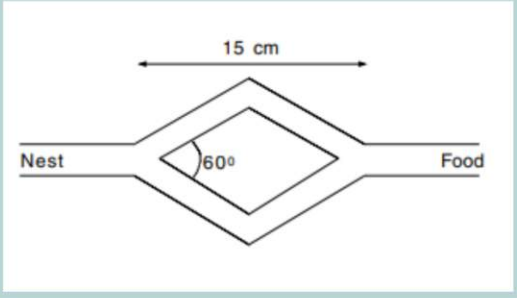
En dos puentes que contectaban una colonia de hormigas y una fuente de comida.

Se ejecuto este experimento con un radio variado $\frac{l_i}{l_s}$, donde l_i es la longitud de la rama **larga** y l_s es la longitud de la rama **corta**.

Primer experimento

En el primer experimento el puente tenia dos ramas con la misma longitud es decir $r = 1$.
Como resultado todas las hormigas resultaron usando el mismo camino.

Figura 3



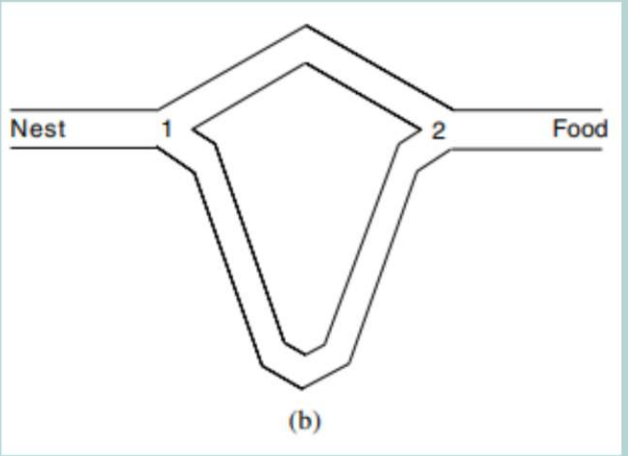
En principio no hay feromonas en ambas ramas, por lo tanto no hay preferencias al seleccionar una ruta, debido a resultados aleatorios mas hormigas escogeran una rama hasta que sea completamente usada.

1 De hormigas reales a artificales

Segundo experimento

En el segundo experimento el radio de la longitud de ambas ramas esta agustado a r = 2, por lo tanto la rama grande sera el doble que la corta.

Figura 4



Como resultado despues de varios experimentos, se vio que las hormigas elegian la rama corta.

Al principio

Las ramas son escogidas de manera aleatoria.

Despues

Las hormigas que escogen la rama corta son las primeras en llegar a la comida y regresar a su colonia.

Por lo tanto

La feoromona empezada a acumularse más rapido en la rama corta, lo cual eventualmente hara que todas las hormigas usen la ruta corta.

1.2 Experimentos de doble puente

El comportamiento de busqueda de comida de las hormigas ha sido investigado en experimentos por varios investigadores.

Por ejemplo

en 1990.

Consistio

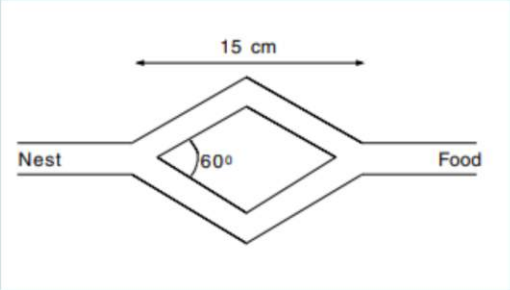
En dos puentes que contectaban una colonia de hormigas y una fuente de comida.

Se ejecuto este experimento con un radio variado $\frac{l_i}{l_s}$, donde l_i es la longitud de la rama y l_s es la longitud de la rama .

Primer experimento

En el primer experimento el puente tenia dos ramas con la misma longitud es decir $r = 1$.
Como resultado todas las hormigas resultaron usando el mismo camino.

Figura 3



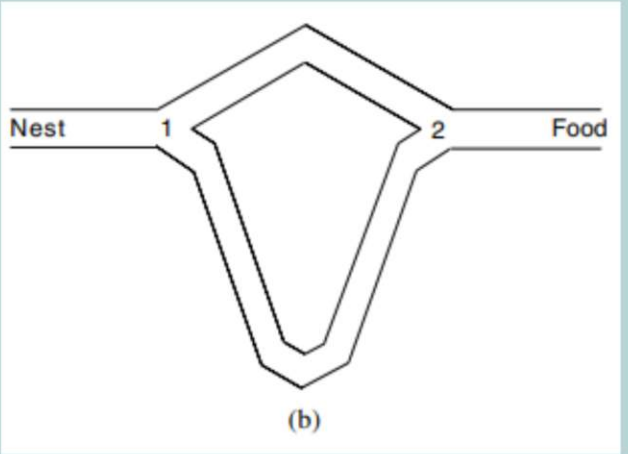
En principio no hay feromonas en ambas ramas, por lo tanto no hay preferencias al seleccionar una ruta, debido a resultados aleatorios mas hormigas escogeran una rama hasta que sea completamente usada.

1 De hormigas reales a artificales

Segundo experimento

En el segundo experimento el radio de la longitud de ambas ramas esta agustado a r = 2, por lo tanto la rama grande sera el doble que la corta.

Figura 4



Como resultado despues de varios experimentos, se vio que las hormigas elegian la rama corta.

Al principio

Las ramas son escogidas de manera aleatoria.

Despues

Las hormigas que escogen la rama corta son las primeras en llegar a la comida y regresar a su colonia.

Por lo tanto

La feoromona empezada a acumularse más rapido en la rama corta, lo cual eventualmente hara que todas las hormigas usen la ruta corta.

1 De hormigas reales a artificales

Autocatálisis

Proceso en el cual a traves de pequeñas interacciones a un nivel "microscopico" produce un patron macroscopico.

Por ejemplo

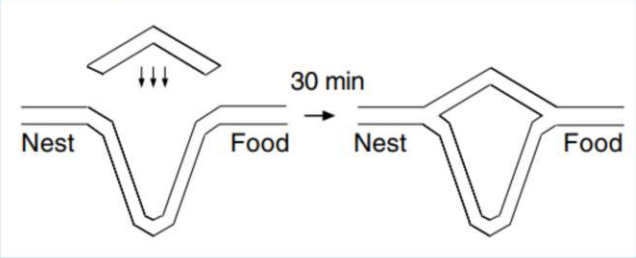
La convergencia a una misma ruta de las hormigas.

Tercer experimento

Es interesante también ver que pasa cuando a una colonia de hormigas que ha alcanzado la convergencia, se le ofrece una nueva ruta más pequeña.

Esta situación fue estudiada en otro experimento, en el cual solo la rama larga fue ofrecida a la colonia y despues de un periodo de 30 minutos la rama corta fue añadida.

Figura 5



Resultado

El resultado fue que la ruta corta fue elegida solo esporadicamente, esto es debido a la gran concentración de feromona en la rama larga y por la lenta evaporación de feromonas.

Evaporación de Feromonas

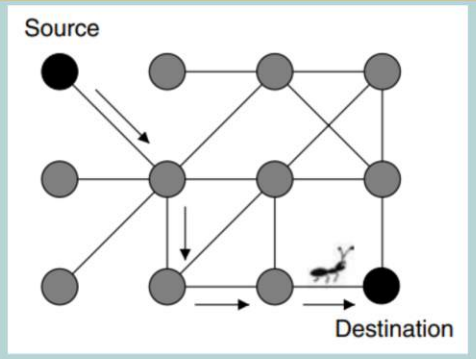
La evaporación es muy lenta para permitir a una colonia de hormigas olvidar, el camino suboptimo al cual han convergido, para que así aprendan una nueva y mejor ruta.

1 De hormigas reales a artificales

1.3 Hormigas artificiales y caminos de costo mínimo

Nuestro objetivo es definir algoritmos que puedan ser usados para resolver problemas de costo mínimo en grafos más complicados.

Figura 6

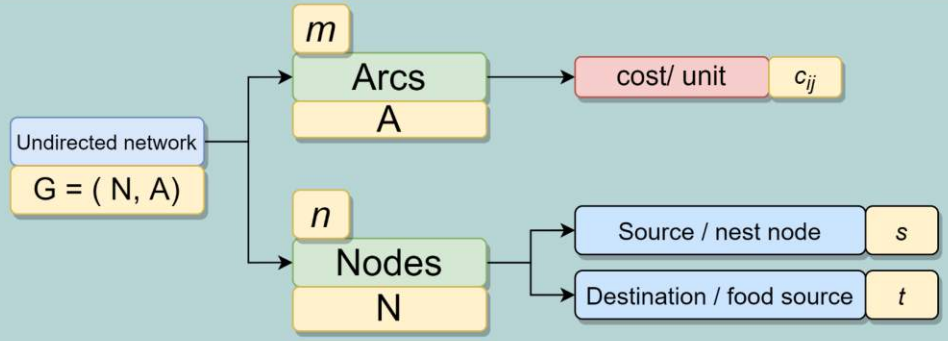


Al principio

Considera un grafo conectado $G = (N, A)$.

Donde

N es el conjunto de ***n*** nodos y A es el conjunto de arcos no dirigidos conectandolos. Los dos puntos que queremos establecer son los nodos source y destination.



Autocatálisis

Proceso en el cual a traves de pequeñas interacciones a un nivel "**microscopico**" produce un patron "**macroscopico**".

Por ejemplo

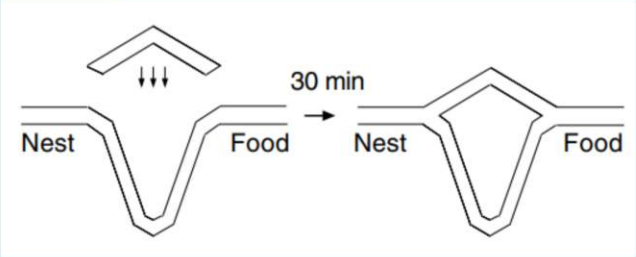
La **convergencia** a una misma ruta de las hormigas.

Tercer experimento

Es interesante también ver que pasa cuando a una colonia de hormigas que ha alcanzado la convergencia, se le ofrece una nueva ruta más pequeña.

Esta situación fue estudiada en otro experimento, en el cual solo la rama larga fue ofrecida a la colonia y despues de un periodo de 30 minutos la rama corta fue añadida.

Figura 5



Resultado

El resultado fue que la ruta corta fue elegida solo **esporadicamente**, esto es debido a la gran concentración de feromona en la rama larga y por la lenta evaporación de feromonas.

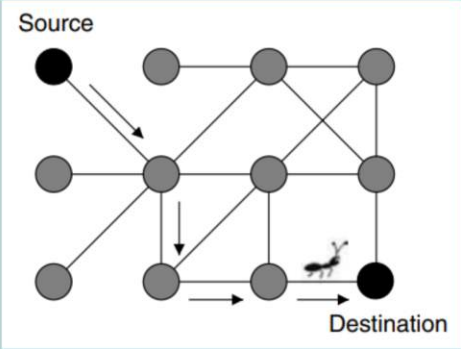
Evaporación de Feromonas

La **evaporación** es muy lenta para permitir a una colonia de hormigas olvidar, el camino suboptimo al cual han convergido, para que así aprendan una nueva y mejor ruta.

1.3 Hormigas artificiales y caminos de costo mínimo

Nuestro objetivo es definir algoritmos que puedan ser usados para resolver problemas de costo mínimo en grafos más complicados.

Figura 6

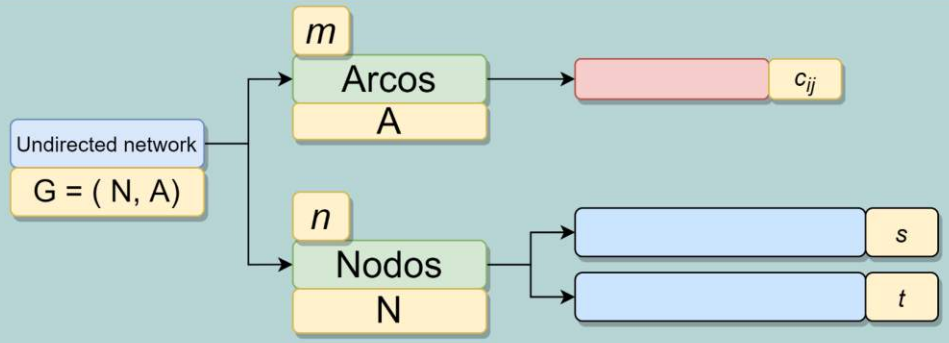


Al principio

Considera un grafo conectado $G = (N, A)$.

Donde

N es el conjunto de ***n*** nodos y A es el conjunto de arcos no dirigidos conectandolos. Los dos puntos que queremos establecer son los nodos source y destination.



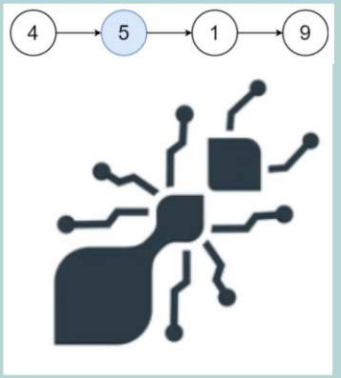
1 De hormigas reales a artificales

Caracteristicas

Debemos establecer algunas caracteristicas para poder establecer hormigas artificiales.

Memoria

Primero las hormigas tienen una forma limitada de memoria en la que guardan las rutas parciales que han seguido hasta el momento.
Asi como el costo de los arcos que han atravesado.



Modos avance y retroceso

Las hormigas tienen dos modos de trabajo: avance y retroceso.

1) Modo de avance

Cuando las hormigas salen del nido hacia la comida, las hormigas en este modo construyen su solucion escogiendo probabilisticamente el siguiente nodo con aquellos nodos vecinos que estan en el grafo.

Probabilidad

Esta eleccion es basada en los senderos de feromonas dejados por otras hormigas en el grafo.

Feromonas

Las hormigas en este modo no depositan feromonas en el camino

1 De hormigas reales a artificales

2) Modo de retroceso

El uso de una memoria permite a la hormiga recordar su camino de regreso.

Feromonas

Mientras las hormigas regresan estas depositan feromonas en las arcos que atraviesan.

3) Actualización de feromonas basado en la calidad de la solución

Las hormigas guardan el costo de los arcos usados.

Por lo tanto

Pueden evaluar el costo de la soluciones que ellas generan y usan esta evaluación para modificar la cantidad de feromonas que depositan en el modo de retroceso.

Esto ayuda

a las hormigas a escoger mejores soluciones.

4) Evaporación de feromonas

Esta caracteristica añada una regla de evaporación a las feromonas, como decreser la cantidad de feromonas en el grafo de una manera constante.

Evaporación

La evaporación de feromonas ayuda a las hormigas a no escoger soluciones de calidad pobre.

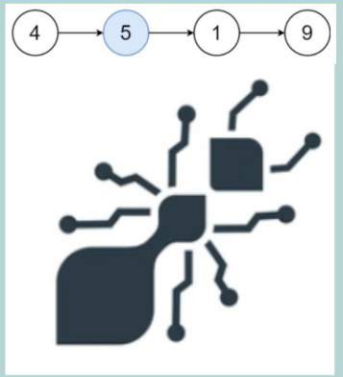
1 De hormigas reales a artificales

Caracteristicas

Debemos establecer algunas caracteristicas para poder establecer hormigas artificiales.

Memoria

Primero las hormigas tienen una forma limitada de _____ en la que guardan las rutas parciales que han seguido hasta el momento.
Asi como el costo de los arcos que han atravesado.



Modos avance y retroceso

Las hormigas tienen dos modos de trabajo: _____ y _____.

1) Modo de avance

Cuando las hormigas salen del nido hacia la comida, las hormigas en este modo _____ su solucion escogiendo probabilisticamente el siguiente nodo con aquellos nodos vecinos que estan en el grafo.

Probabilidad

Esta eleccion es basada en los senderos de feromonas dejados por otras hormigas en el grafo.

Feromonas

Las hormigas en este modo no depositan feromonas en el camino

1 De hormigas reales a artificales

2) Modo de retroceso

El uso de una memoria permite a la hormiga _____ su camino de regreso.

Feromonas

Mientras las hormigas regresan estas depositan feromonas en las arcos que atraviesan.

3) Actualización de feromonas basado en la calidad de la solución

Las hormigas guardan el costo de los arcos usados.

Por lo tanto

Pueden evaluar el costo de la soluciones que ellas generan y usan esta evaluación para _____ la cantidad de feromonas que depositan en el modo de retroceso.

Esto ayuda

a las hormigas a escoger mejores soluciones.

4) Evaporación de feromonas

Esta caracteristica añada una regla de evaporación a las feromonas, como _____ la cantidad de feromonas en el grafo de una manera constante.

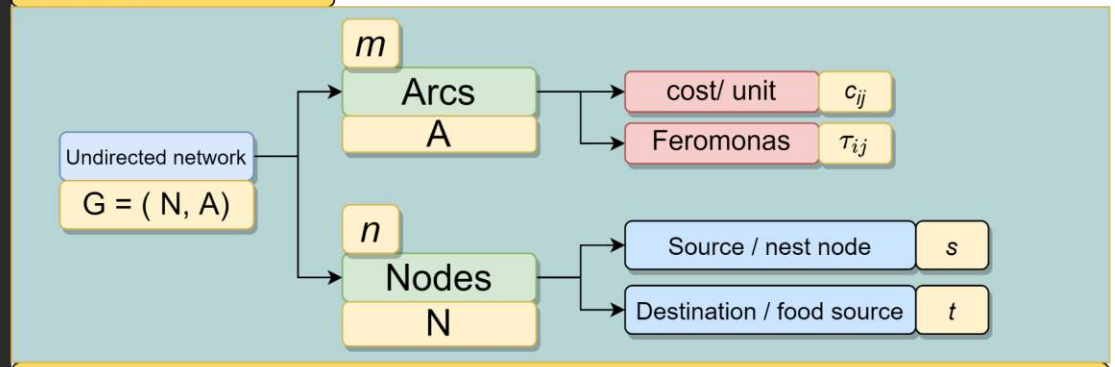
Evaporación

La evaporación de feromonas ayuda a las hormigas a no escoger soluciones de calidad pobre.

S-ACO

Ahora presentaremos los detalles de los algoritmos S-ACO

Modelo



Feromona de sendero artificial

Para cada arco (i,j) en el grafo G = (N,A) asociamos una variable τ_{ij} llamada feromona de sendero artificial, las cuales son leídas y escritas por las hormigas.

Comportamiento de las hormigas al buscar su camino

Cada hormiga construye su solución empezando desde el source node, en cada paso se usa información local guardad para decidir de una manera estocastica el siguiente nodo.

1 De hormigas reales a artificales

Principio

Al principio hay una cantidad de feromonas constante $\tau_{ij} = 1, \forall (i,j) \in A$ para todos los arcos.

Escojer j como nodo

Cuando el nodo esta en i una hormiga k usa la feromona τ_{ij} para calcular la probabilidad de escoger j.

$$p_{ij}^k = \frac{(\tau_{ij})^\alpha}{\sum_{l \in N_i^k} (\tau_{il})^\alpha}, \quad j \in N_{ij}^k$$

Donde N_i^k es el vecindario de la hormiga k en el nodo i; estos son todos los nodos que estan directamente conectados al nodo i, a excepcion del predecesor del nodo i.

De esta manera las hormigas evaden regresarse por el mismo que camino que llegaron.

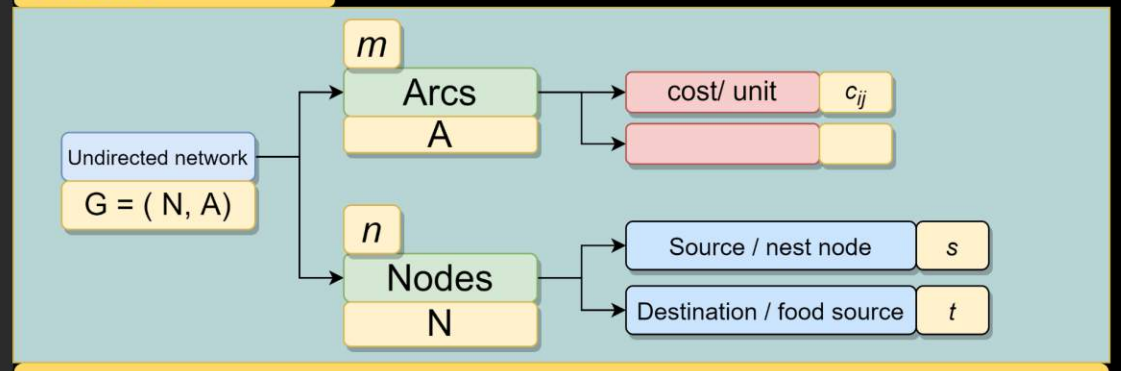
N_i^k esta vacío

En caso de que N_i^k este vacio es decir un lugar sin camino el predecesor de i es incluido en N_i^k.

S-ACO

Ahora presentaremos los detalles de los algoritmos S-ACO

Modelo



Feromona de sendero artificial

Para cada arco (i,j) en el grafo G = (N,A) asociamos una variable llamada feromona de sendero artificial, las cuales son leídas y escritas por las hormigas.

Comportamiento de las hormigas al buscar su camino

Cada hormiga construye su solución empezando desde el node, en cada paso se usa información local guardad para decidir de una manera estocastica el siguiente nodo.

Principio

Al principio hay una cantidad de feromonas constante $\tau_{ij} =$, $\forall (i,j) \in A$ para todos los arcos.

Escojer j como nodo

Cuando el nodo esta en i una hormiga k usa la feromona τ_{ij} para calcular la probabilidad de escoger j.

$$p_{ij}^k = \qquad , \quad j \in N_{ij}^k$$

Donde N_i^k es el de la hormiga k en el nodo i; estos son todos los nodos que estan directamente conectados al nodo i, a excepcion del predecesor del nodo i.

De esta manera las hormigas evaden regresarse por el mismo que camino que llegaron.

N_i^k esta vacío

En caso de que N_i^k este vacio es decir un lugar sin camino el predecesor de i es incluido en N_i^k.

7.1.2 Resultados experimentales y aplicaciones del mundo real

Como hemos visto ACO ha sido experimentado en muchos problemas academicos

Tales como

Problema del viajero, problemas de asignación, problemas de programación, subconjunto y problemas de satisfacción de restricciones.

ACO vs otros

Para muchos de los problemas anteriores, se ha alcanzado rendimiento de clase **mundial.**

Estado del arte

En este momento su rendimiento es comparable o **mejo**r a problemas:

A	**orden secuencial**	**Gambardella, 2000**
B	**Ruteo de vehiculos con tiempo**	**Gambardella, 1999**
C	**Asignación cuadratica**	**Maniezo, 1999**
A	**Programacion de tienda grupal**	**Blum, 2003**
B	**Arbol de cardinalidad i con peso en arcos**	**Blum, 2003**

Otros algoritmos

AntNet ha sido muy eficiente en problemas de ruteo en redes.

Aplicaciones mundo real

A	**EuroBros**	ACO aproblemas de scheduling
B	**AntoOptima**	Aco a problema de ruteo de vehiculos
B	**Pina Petroli**	Dyvoil y AntRoute, para distribución de petroleo

7.2 Tendencias actuales de ACO

Introducción

Hoy en día cientos de articulos han sido escritos sobre aplicaciones de ACO.

Sin embargo

Hay varias areas en donde solo se han dado pasos **preliminares** y en donde mucha investigación se necesita.

Posibles areas de investigación

1 Problemas dinámicos

En estos los datos de la instancia, tales como los valores objetivo, parametros de descición o restricciones, pueden cambiar mientras se reselve el problema.

2 Problemas estocasticos

En el que uno solo tiene información probabilistica sobre los valores de la función objetivo.

3 Problemas multiobjetivo

En el cual una función objetivo multiple evalua criterios competitivos para la calidad de la solución.

Paralelización de ACO

La investigación activa tambien esta trabajando en una **paralelización** efectiva de ACO.

7.1.2 Resultados experimentales y aplicaciones del mundo real

Como hemos visto ACO ha sido experimentado en muchos problemas academicos

Tales como

Problema del viajero, problemas de asignación, problemas de programación, subconjunto y problemas de satisfacción de restricciones.

ACO vs otros

Para muchos de los problemas anteriores, se ha alcanzado rendimiento de clase _____.

Estado del arte

En este momento su rendimiento es comparable o _____ a problemas:

A	orden secuencial	Gambardella, 2000
B	Ruteo de vehiculos con tiempo	Gambardella, 1999
C	Asignación cuadratica	Maniezo, 1999
A	Programacion de tienda grupal	Blum, 2003
B	Arbol de cardinalidad i con peso en arcos	Blum, 2003

Otros algoritmos

_____ ha sido muy eficiente en problemas de ruteo en redes.

Aplicaciones mundo real

A	EuroBros	ACO aproblemas de scheduling
B	AntoOptima	Aco a problema de ruteo de vehiculos
B	Pina Petroli	Dyvoil y AntRoute, para distribución de petroleo

7.2 Tendencias actuales de ACO

Introducción

Hoy en día cientos de articulos han sido escritos sobre aplicaciones de ACO.

Sin embargo

Hay varias areas en donde solo se han dado pasos _____ y en donde mucha investigación se necesita.

Posibles areas de investigación

1 Problemas dinámicos

En estos los datos de la instancia, tales como los valores objetivo, parametros de descición o restricciones, pueden cambiar mientras se reselve el problema.

2 Problemas estocasticos

En el que uno solo tiene información probabilistica sobre los valores de la función objetivo.

3 Problemas multiobjetivo

En el cual una función objetivo multiple evalua criterios competitivos para la calidad de la solución.

Paralelización de ACO

La investigación activa tambien esta trabajando en una _____ efectiva de ACO.

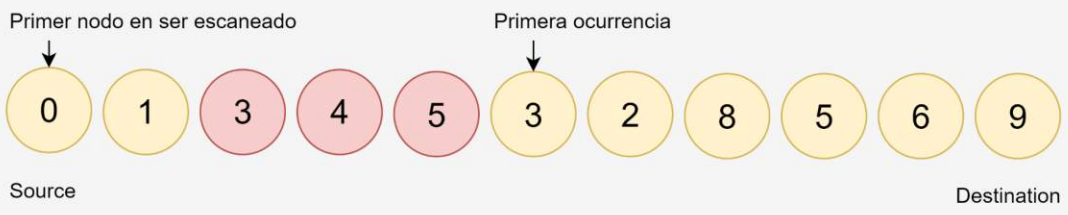
1.3.1 Retroceso de ruta y actualización de feromonas

Una vez que la hormiga llega al nodo destino, la hormiga cambia del modo avance al modo retroceso siguiendo el camino que tomo hasta el nido.

Una funcionalidad adicional es que antes de empezar el camino de regreso la hormiga elimina cualquier **ciclo** generado en su ruta.

Eliminar rutas

Esto puede hacerse escaneando iterativamente la posición de un nodo en en su índice i, empezando desde el nodo inicial.



Los ciclos son eliminados en el mismo orden que fueron creados

Algoritmo

```
function RemoverCiclos(Ruta)
    NRuta = [];
    N = Ruta.size();
    for (int i = 0; i < N; i++)
        for (int j = i + 1; j < N; j++)
            if (Ruta[i] == Ruta[j])
                i = j;
    NRuta.Add(Ruta[i]);
```

Pasamos por cada nodo una sola vez

Iniciamos j con el siguiente nodo despues de i,
Si el nodo es igual a j entonces brincamos a este.
Añadimos el nodo a la nueva ruta.

1.3.1 Retroceso de ruta y actualización de feromonas

Durante su viaje de regreso una hormiga cualquiera k deposita una cantidad $\Delta\tau^k$ de feromona en los arcos que ha visitado.
Y cambia el valor de la feromona τ_{ij} de la siguiente manera

$$\tau_{ij} = \tau_{ij} + \Delta\tau^k$$

Las hormigas depositan esta cantidad de feromonas en función de la longitud de la ruta, entre más corto el camino mayor la cantidad de feromonas.

Evaporación de feromonas

Debido ala complejidad de problemas que tienen que ser resultos por una colonia de hormigas artificial, se ha tenido que añadir un sistema para olvidar desciciones a soluciones pobres.

Este sistema se llama **evaporacion de fermonas**.

Este proceso esta intercalado con el deposito de feromonas y se aplica de la siguiente manera a todos los arcos.

$$\tau_{ij} = (1-\rho)\tau_{ij} \quad ,\forall(i,j) \in A$$

donde $\rho \in (0,1]$, despues de que la feromona haya sido aplicada a todos los arcos, la feromona $\Delta\tau^k$ es añadida.

Decimos que una ireación de S-ACO completa, involucra al movimiento de las hormigas, evaporación de las feromonas y al deposito de fermonas.

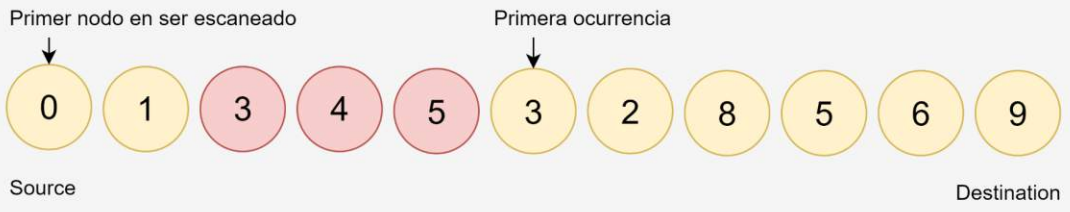
1.3.1 Retroceso de ruta y actualización de feromonas

Una vez que la hormiga llega al nodo destino, la hormiga cambia del modo avance al modo retroceso siguiendo el camino que tomo hasta el nido.

Una funcionalidad adicional es que antes de empezar el camino de regreso la hormiga elimina cualquier _____ generado en su ruta.

Eliminar rutas

Esto puede hacerse escaneando iterativamente la posición de un nodo en en su índice i, empezando desde el nodo inicial.



Los ciclos son eliminados en el mismo orden que fueron creados

Algoritmo

```
function RemoverCiclos(Ruta)
    NRuta = [];
    N = Ruta.size();
    for (int i = 0; i < N; i++)
        for (int j = i + 1; j < N; j++)
            if (Ruta[i] == Ruta[j])
                i = j;
    NRuta.Add(Ruta[i]);
```

Pasamos por cada nodo una sola vez

Iniciamos j con el siguiente nodo despues de i,
Si el nodo es igual a j entonces brincamos a este.
Añadimos el nodo a la nueva ruta.

1.3.1 Retroceso de ruta y actualización de feromonas

Durante su viaje de regreso una hormiga cualquiera k deposita una cantidad $\Delta\tau^k$ de feromona en los arcos que ha visitado.
Y cambia el valor de la feromona τ_{ij} de la siguiente manera

$$\tau_{ij} = \quad$$

Las hormigas depositan esta cantidad de feromonas en función de la longitud de la ruta, entre más corto el camino mayor la cantidad de feromonas.

Evaporación de feromonas

Debido ala complejidad de problemas que tienen que ser resultos por una colonia de hormigas artificial, se ha tenido que añadir un sistema para olvidar desciciones a soluciones pobres.

Este sistema se llama _____.

Este proceso esta intercalado con el deposito de feromonas y se aplica de la siguiente manera a todos los arcos.

$$\tau_{ij} = \quad , \forall (i,j) \in A$$

donde $\rho \in (0,1]$, despues de que la feromona haya sido aplicada a todos los arcos, la feromona $\Delta\tau^k$ es añadida.

Decimos que una ireación de S-ACO completa, involucra al movimiento de las hormigas, evaporación de las feromonas y al deposito de fermonas.

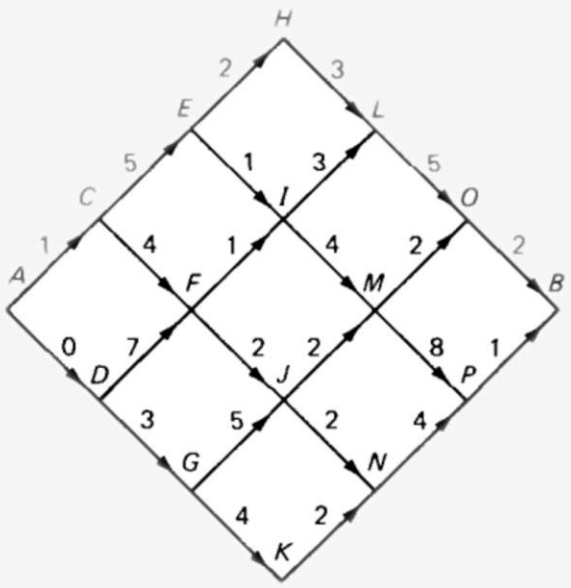
Programación dinámica

1 Problemas de ruta elemental

Programación dinámica es un proceso de optimización que es aplicable a problemas que requieren una secuencia de desiciones.

1.2 Problemas de ruta simple

Supon que vives en una ciudad que organiza sus calles como la siguiente imagen



Tu vives en A y deseas llegar a B en la menor distancia posible, ¿Cómo lo harias?

Opción Enumeración

Podemos usar por supuesto la enumeración de todas las posibles rutas de **A** hasta **B** y despues escogiendo la menor de la sumatoria de sus distancias, sin embargo uno puede resolver este problema con un método más eficiente, este siendo programación dinámica.

1.3 Solución programación dinámica

La programación dinámica desarrolla su solución de la siguiente manera

Yo no se ir hacia abajo o hacia arriba desde el punto A, Sin embargo si de alguna manera supiera el costo requerido para llegar de C a B en su mejor ruta y tambien supiera el costo requerido para llegar de D a B en su mejor ruta, yo puedo tomar la mejor solución en A.

Denotamos el costo mínimo de C a B como S_c y denotamos el costo mínimo desde D a B como S_D

Despues añadiriamos C_{AD} a S_D y C_{AC} a S_C, y compararíamos estas dos sumas para escoger el mejor.

Desafortunadamente sabemos que no conocemos todavia los dos números S_C y S_D.

Sin embargo una de las dos ideas clave en la programación dinámica ya se ha mostrado, y esta es la observación de que solo los esfuerzos a traves de la mejor ruta desde C y D a B son relevantes.

Esta observación es llamada **principio de óptimalidad**.

La ruta óptima de A a B, tiene la propiedad, de que los siguientes puntos en la ruta después de B, tambien son la ruta óptima a B.

Habiendo denotado S_C y S_D podemos citar el principio con la formula siguiente:

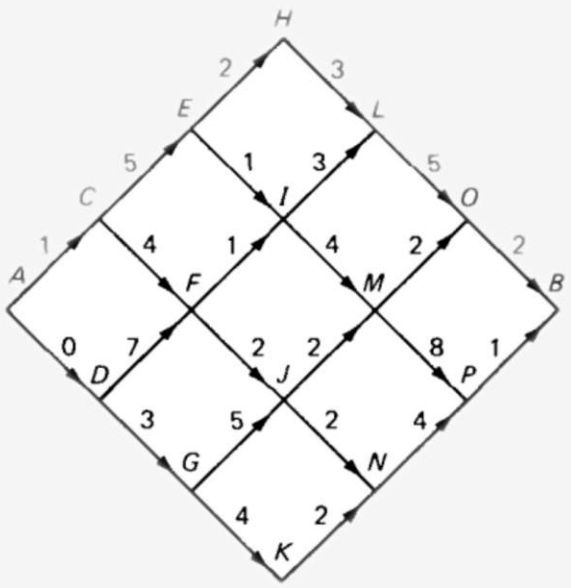
$$S_A = min \begin{cases} 1 & + S_C \\ 0 & + S_D \end{cases}$$

1 Problemas de ruta elemental

Programación dinámica es un proceso de optimización que es aplicable a problemas que requieren una _____.

1.2 Problemas de ruta simple

Supon que vives en una ciudad que organiza sus calles como la siguiente imagen



Tu vives en A y deseas llegar a B en la menor distancia posible, ¿Cómo lo harias?

Opción Enumeración

Podemos usar por supuesto la enumeración de todas las posibles rutas de **A** hasta **B** y despues escogiendo la menor de la sumatoria de sus distancias, sin embargo uno puede resolver este problema con un método más eficiente, este siendo programación dinámica.

1.3 Solución programación dinámica

La programación dinámica desarrolla su solución de la siguiente manera

Yo no se ir hacia abajo o hacia arriba desde el punto A, Sin embargo si de alguna manera supiera el costo requerido para llegar de C a B en su mejor ruta y tambien supiera el costo requerido para llegar de D a B en su mejor ruta, yo puedo tomar la mejor solución en A.

Denotamos el costo mínimo de C a B como S_c y denotamos el costo mínimo desde D a B como S_D

Despues añadiriamos C_{AD} a S_D y C_{AC} a S_C, y compararíamos estas dos sumas para escoger el mejor.

Desafortunadamente sabemos que no conocemos todavia los dos números S_C y S_D.

Sin embargo una de las dos ideas clave en la programación dinámica ya se ha mostrado, y esta es la observación de que solo los esfuerzos a traves de la mejor ruta desde C y D a B son relevantes.

Esta observación es llamada _____.

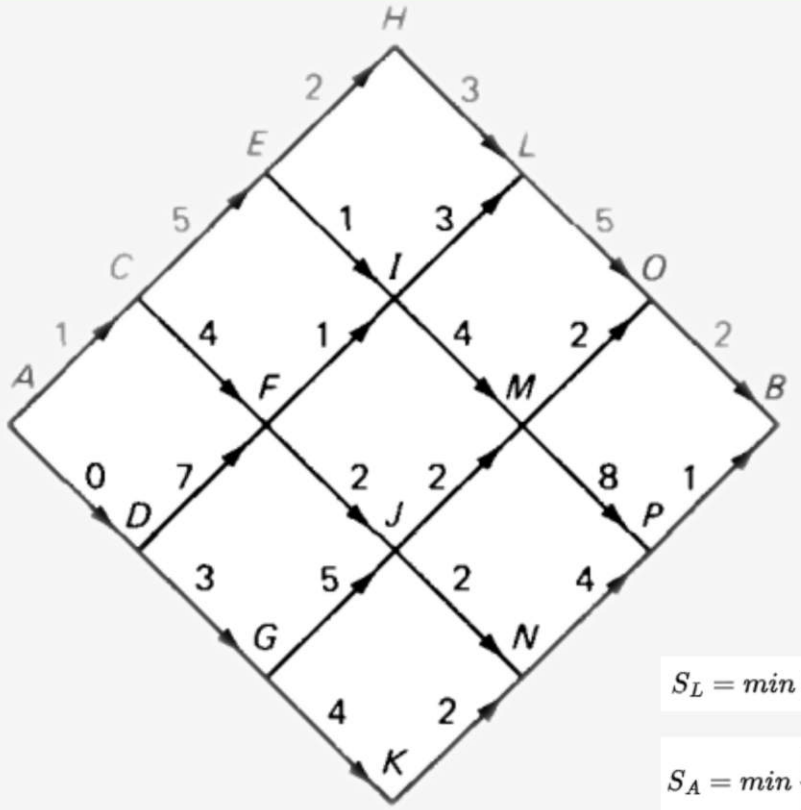
La ruta óptima de A a B, tiene la propiedad, de que los siguientes puntos en la ruta después de B, tambien son la ruta óptima a B.

Habiendo denotado S_C y S_D podemos citar el principio con la formula siguiente:

1

Ejercicio

Resolver el siguiente modelo, Ruta optima de S_A



$$S_A = min\begin{cases}1 & + S_C \\ 0 & + S_D\end{cases} = 13$$

$$S_H = min\{3 \quad + S_L = 10$$

$$S_L = min\{5 \quad + S_O \quad = 7$$

$$S_I = min\begin{cases}3 & + S_L \\ 4 & + S_M\end{cases} = 8$$

$$S_E = min\begin{cases}2 & + S_H \\ 1 & + S_I\end{cases} = 9$$

$$S_C = min\begin{cases}5 & + S_E \\ 4 & + S_F\end{cases} = 12$$

$$S_A = min\begin{cases}2 & + S_O \\ 8 & + S_P\end{cases} = 4$$

$$S_F = min\begin{cases}1 & + S_I \\ 2 & + S_J\end{cases} = 8$$

$$S_D = min\begin{cases}7 & + S_F \\ 3 & + S_G\end{cases} = 14$$

$$S_J = min\begin{cases}2 & + S_M \\ 2 & + S_N\end{cases} = 6$$

$$S_G = min\begin{cases}5 & + S_J \\ 4 & + S_K\end{cases} = 11$$

$$S_N = min\{4 \quad + S_P = 5$$

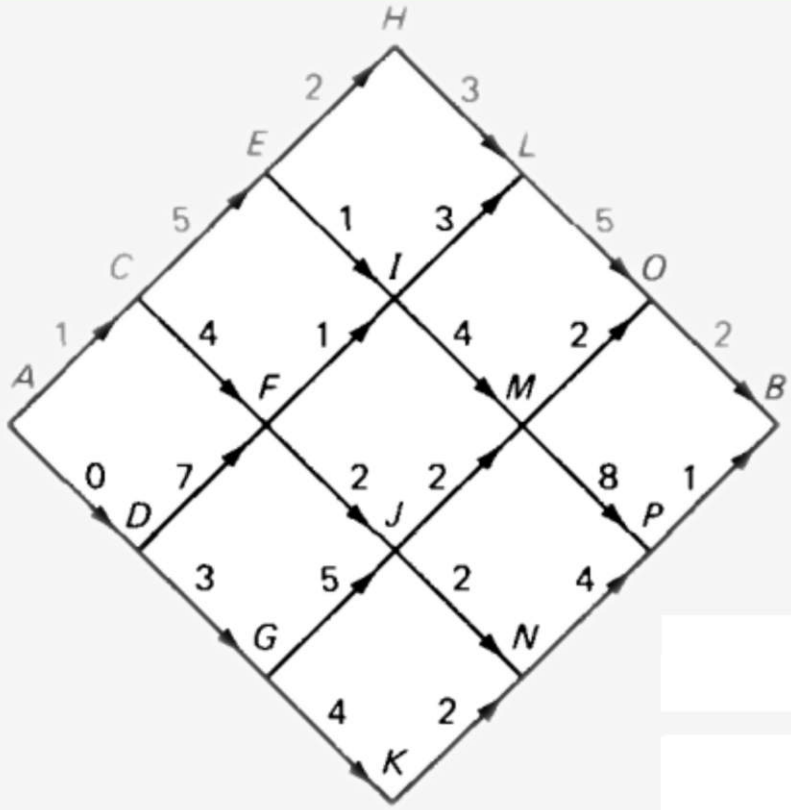
$$S_K = min\{2 \quad + S_N = 7$$

Esfuerzo computacional total: 24 adiciones y 9 comparaciones

1

Ejercicio

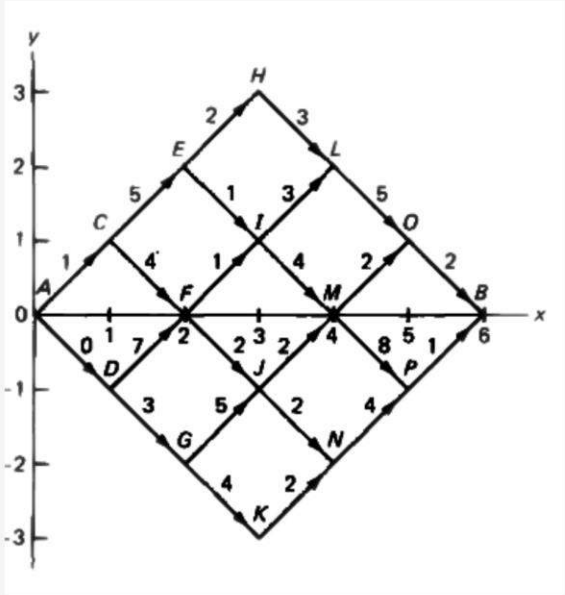
Resolver el siguiente modelo, Ruta optima de S_A



Esfuerzo computacional total:

1 Terminología

Ahora desarrollaremos una notación particular para el problema de ruta simple, el cual nos permitira tener una representación más sistematisada. Pongamos nuestra ciudad, pero ahora con un sistema de coordenadas.



Ahora el punto A tiene las coordenadas (0,0) y el punto B tiene las coordenadas (6,0).

La función de optimalidad es ahora una función con un par x y y

$S(x,y) =$ *el valor de la ruta mínima que conecta el vertice (x,y) con el vertice (6,0).*

Denotamos al simbolo $a_u(x,y)$ como la distancia asociada al arco que conecta el vertice (x,y) con el vértice (x+1, y+1).

u representa la dirección que lleva nuestro arco siendo u hacia arriba y d hacia abajo a_u, a_d.

Arco infinito

Decimos que $a_u(x,u)$ o $a_d(x,y)$ es igual a ∞, si no existe este en nuestra red, por ejemplo $a_u(4,2) = \infty$

Relación de recurrencia

En estos nuevos terminos el principio de óptimalidad da la siguiente relacion de recurrencia:

$$S(x,y) = \min \begin{bmatrix} a_u(x,y) + S(x+1, y+1) \\ a_d(x,y) + S(x+1, y-1) \end{bmatrix}$$

Y la condición frontera

$$S(6,0) = 0,$$

En los siguientes ejercicios cuando se use la frase "da la formulación de programación dinámica a este problema" nos referiremos a:

1 Define la función de valor óptima apropidada, incluyendo la definición especifica de sus argumentos y el significado del valor de la función

$S(x,y) =$ *el valor de la ruta con menor distancia conectando al vertice (x,y) con otro vertice (6,0).*

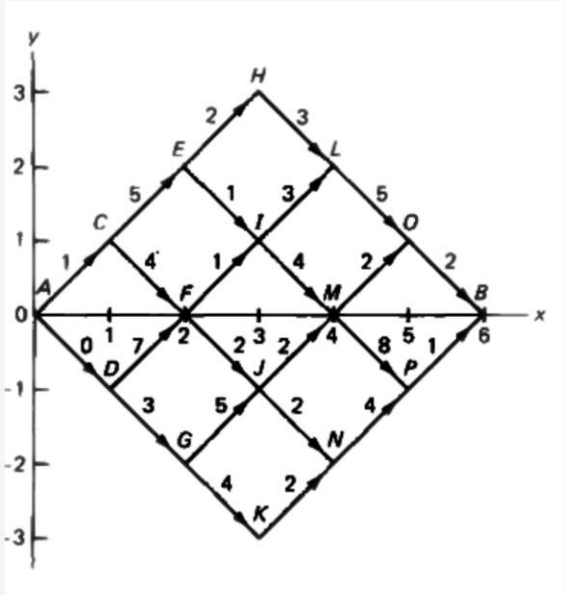
2 Escribe la relación de recurrencia apropiada

$$S(x,y) = \min \begin{bmatrix} a_u(x,y) + S(x+1, y+1) \\ a_d(x,y) + S(x+1, y-1) \end{bmatrix}$$

3 Escribe las condiciones de frontera apropiadas

1 Terminología

Ahora desarrollaremos una notación particular para el problema de ruta simple, el cual nos permitira tener una representación más sistematisada. Pongamos nuestra ciudad, pero ahora con un .



Ahora el punto A tiene las coordenadas y el punto B tiene las coordenadas .

La función de optimalidad es ahora una función con un par x y y

$S(x,y) =$ *el valor de la ruta mínima que conecta el vertice (x,y) con el vertice (6,0).*

Denotamos al simbolo $a_u(x,y)$ como la distancia asociada al arco que conecta el vertice (x,y) con el vértice (x+1, y+1).

u representa la dirección que lleva nuestro arco siendo ***u*** hacia arriba y ***d*** hacia abajo a_u, a_d.

Arco infinito

Decimos que $a_u(x,u)$ o $a_d(x,y)$ es igual a , si no existe este en nuestra red, por ejemplo $a_u(4,2) =$

Relación de recurrencia

En estos nuevos terminos el principio de óptimalidad da la siguiente relacion de recurrencia:

$$S(x,y) = \min \begin{bmatrix} a_u(x,y) + S(x+1, y+1) \\ a_d(x,y) + S(x+1, y-1) \end{bmatrix}$$

Y la condición frontera

En los siguientes ejercicios cuando se use la frase "da la formulación de programación dinámica a este problema" nos referiremos a:

1 Define la función de valor óptima apropidada, incluyendo la definición especifica de sus argumentos y el significado del valor de la función

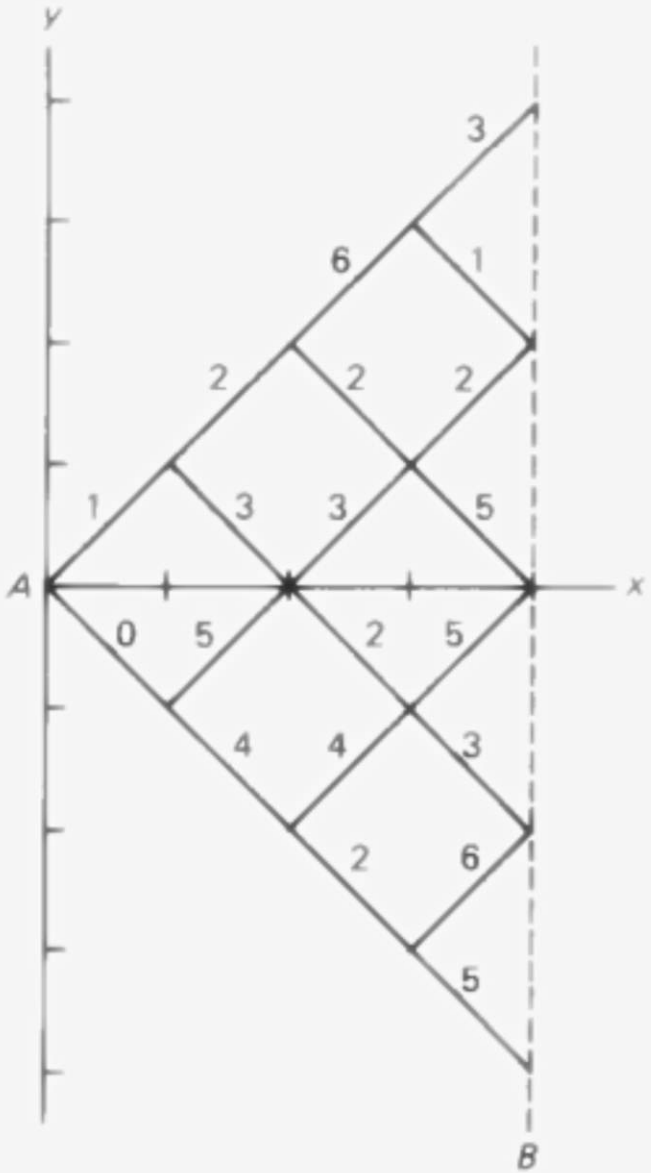
2 Escribe la relación de recurrencia apropiada

3 Escribe las condiciones de frontera apropiadas

1.2

Ejercicio

Buscamos la ruta que conecte al punto A con cualquier punto en la linea B, el cual minimize la sumatoria de los cuadtro artos encontrados, a travez de la ruta. Da la formulacion de programación dinámica de este problema.



Función de valor óptima:

S(0,0) = al valor de la ruta óptima del punto (0,0) a cualquier punto (4,y).

Relación de recurrencia:

$$S(x,y) = min \begin{cases} a_U(x,y) & + S(x+1,y+1) \\ a_D(x,y) & + S(x+1,y-1) \end{cases}$$

Condición de frontera:

S(4,y) = 0

Solución: 7

$$S(3,3) = min \begin{cases} 3 \\ 1 \end{cases} = 1$$

$$S(3,1) = min \begin{cases} 2 \\ 5 \end{cases} = 2$$

$$S(3,-1) = min \begin{cases} 5 \\ 3 \end{cases} = 3$$

$$S(3,-3) = min \begin{cases} 6 \\ 5 \end{cases} = 5$$

$$S(2,2) = min \begin{cases} 6+1 \\ 2+2 \end{cases} = 4$$

$$S(2,0) = min \begin{cases} 3+2 \\ 2+3 \end{cases} = 5$$

$$S(2,-2) = min \begin{cases} 4+3 \\ 2+5 \end{cases} = 7$$

$$S(1,1) = min \begin{cases} 2+4 \\ 3+5 \end{cases} = 6$$

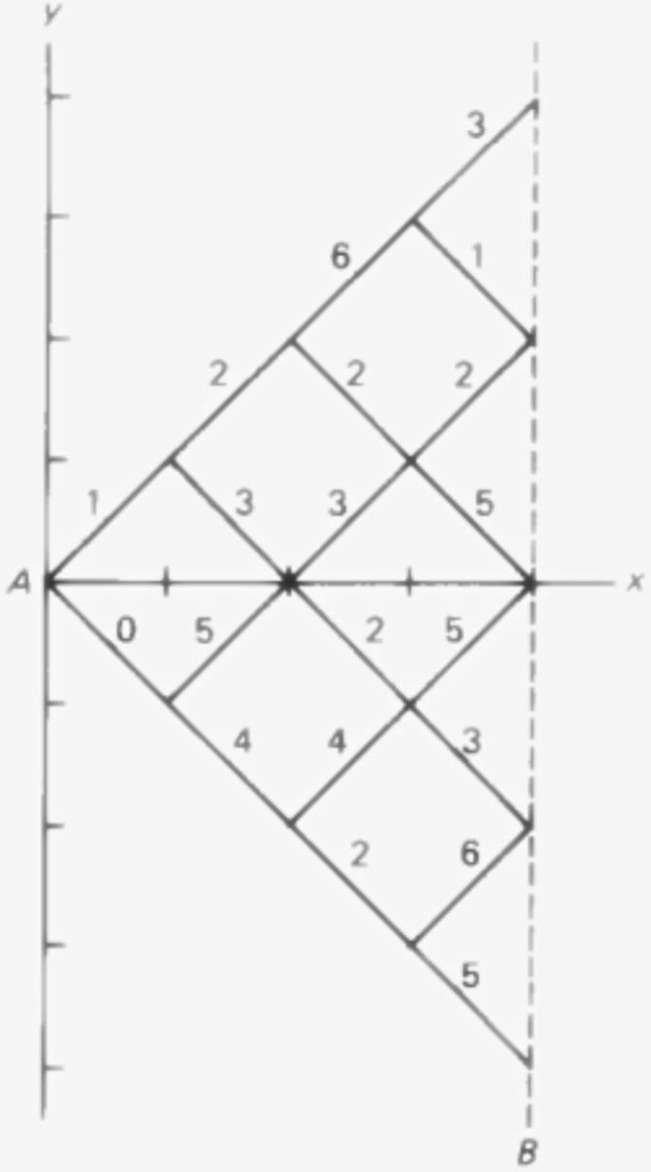
$$S(1,-1) = min \begin{cases} 5+5 \\ 4+7 \end{cases} = 10$$

$$S(0,0) = min \begin{cases} 1+6 \\ 0+10 \end{cases} = 7$$

1.2

Ejercicio

Buscamos la ruta que conecte al punto A con cualquier punto en la linea B, el cual minimize la sumatoria de los cuadtro artos encontrados, a travez de la ruta. Da la formulacion de programación dinámica de este problema.



Función de valor óptima:

Relación de recurrencia:

Condición de frontera:

Solución:

1.6 Programación dinámica de avance

Ahora explicaremos una variación del procedimiento de programación dinámica, de cierta manera invertimos todo nuestro pensamiento, y definimos nuestra función de valor óptimo como

S(x,y) = al valor de la ruta más corta que conecta el vértice inicial (0,0) a el vertice (x,y).

Nuestra relación de recurrencia queda como

$$S(x, y) = \min \begin{bmatrix} a_u(x-1, y-1) + S(x-1, y-1) \\ a_d(x-1, y+1) + S(x-1, y+1) \end{bmatrix}.$$

Condición de frontera

$$S(0, 0) = 0$$

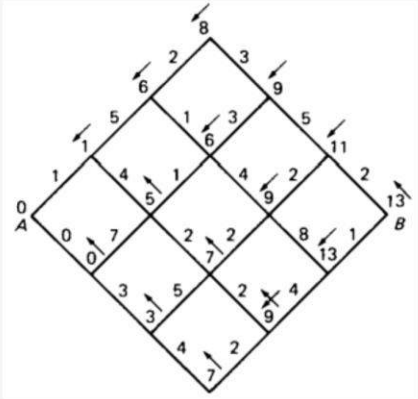
Llamaremos al proceso usando el punto de vista invertido como:

Procedimiento de programación dinámica de avance

Diferencias

Los 2 procesos difieren en la información auxiliar que producen.

El procedimiento de avance, nos da el camino óptimo de A a cada vértice, pero no nos dice nada sobre las rutas óptimas a B.



Problema 1.5

Has el siguiente problema por método de avance, calcula y compara el número de adiciones y comparaciones para el de avance y retroceso.

Avance

Función de valor óptima:

s(x,y) = El valor de la ruta más corta del (x,y) al punto (4,y)

Relación de recurrencia:

$$S(x,y) = min \begin{cases} a_u(x,y) & + S(x+1,y+1) \\ a_d(x,y) & + S(x+1,y-1) \end{cases}$$

Condición de frontera:

$$S(4,y) = 0$$

Solución avance:	7
Comparaciones	10
Adiciones	20

Retraso

Función de valor óptima:

s(x,y) = El valor de la ruta más corta del (x,y) al punto (0,0)

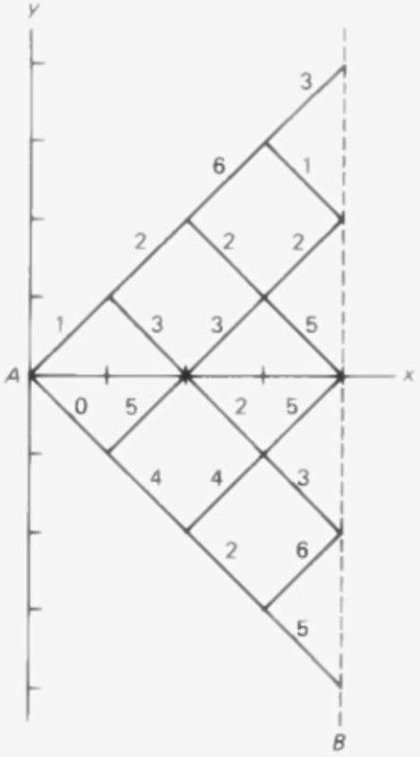
Relación de recurrencia:

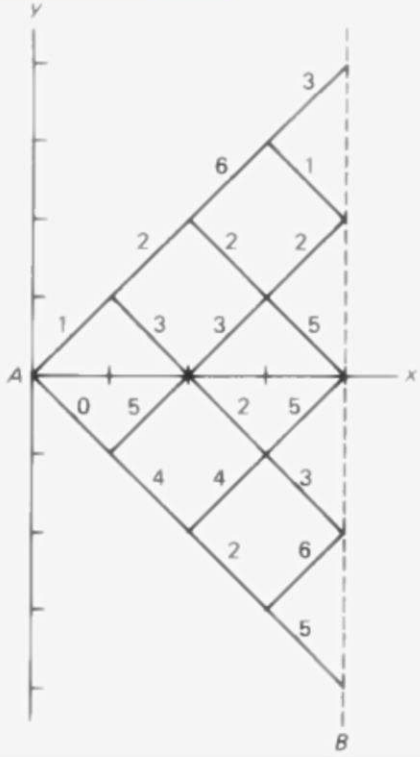
$$S(x,y) = min \begin{cases} a_u(x-1,y-1) & + S(x-1,y-1) \\ a_d(x+1,y+1) & + S(x-1,y+1) \end{cases}$$

Condición de frontera:

$$S(0,0) = 0$$

Solución retroceso:	7
Comparaciones	10
Adiciones	20





1.6 Programación dinámica de avance

Ahora explicaremos una variación del procedimiento de programación dinámica, de cierta manera invertimos todo nuestro pensamiento, y definimos nuestra función de valor óptimo como

S(x,y) = al valor de la ruta más corta que conecta el vértice inicial (0,0) a el vertice (x,y).

Nuestra relación de recurrencia queda como

$$S(x, y) = \min \begin{bmatrix} a_u(x-1, y-1) + S(x-1, y-1) \\ a_d(x-1, y+1) + S(x-1, y+1) \end{bmatrix}.$$

Condición de frontera

$$S(0, 0) = 0$$

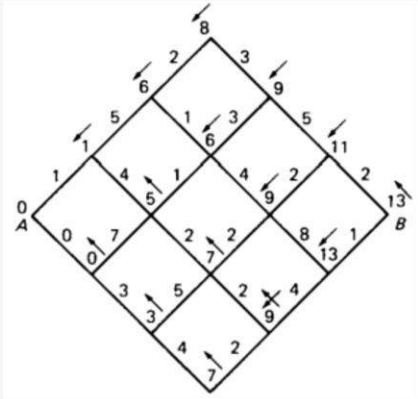
Llamaremos al proceso usando el punto de vista invertido como:

Procedimiento de programación dinámica de avance

Diferencias

Los 2 procesos difieren en la información auxiliar que producen.

El procedimiento de avance, nos da el camino óptimo de A a cada vértice, pero no nos dice nada sobre las rutas óptimas a B.

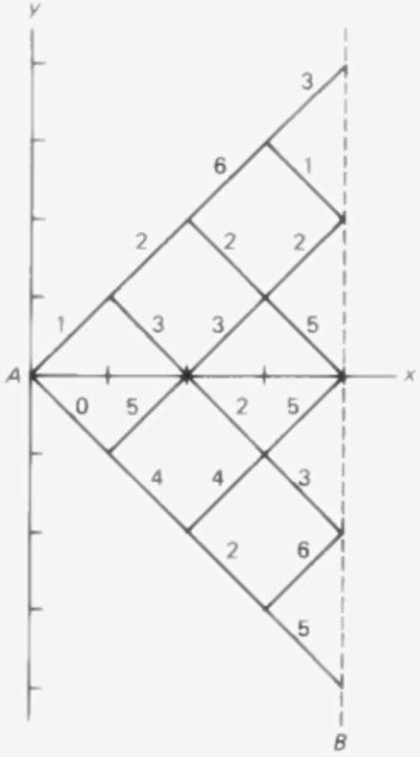


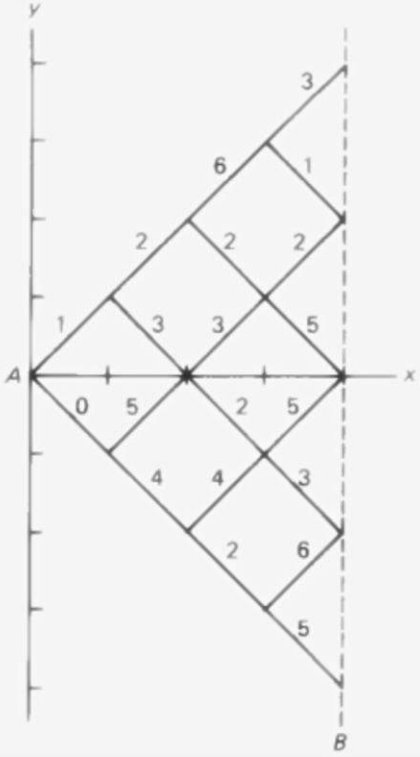
Problema 1.5

Has el siguiente problema por método de avance, calcula y compara el número de adiciones y comparaciones para el de avance y retroceso.

Avance	
Función de valor óptima:	
Relación de recurrencia:	
Condición de frontera:	
Solución avance:	
Comparaciones	
Adiciones	

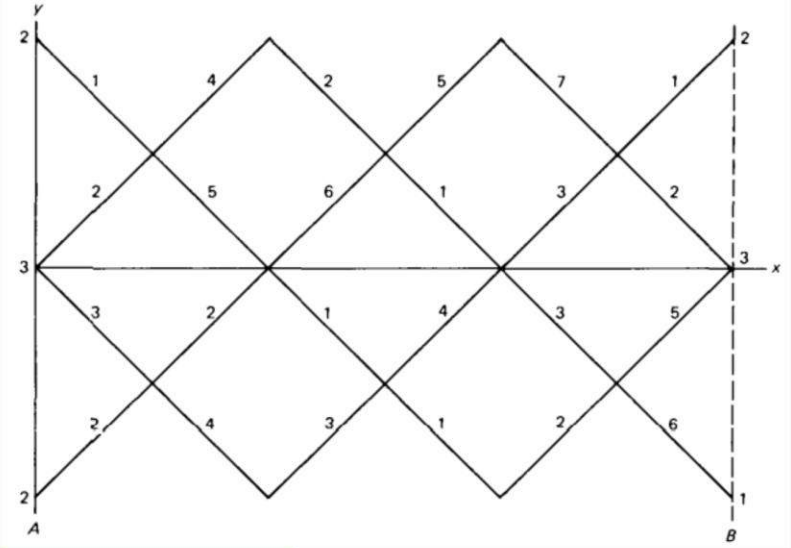
Retraso	
Función de valor óptima:	
Relación de recurrencia:	
Condición de frontera:	
Solución retroceso:	
Comparaciones	
Adiciones	





Problema 1.6

Encuentra el valor de la ruta de costo mínimo desde la linea A hasta la B. los números mostrados a través de las lineas A y B son costos adicionales asociados con varios puntos iniciales y terminales.



Función de valor óptima:

S(x,y) = el valor de la ruta de costo mínimo desde (x,y) hasta el punto (6,y).

Relación de recurrencia:

$$s(x,y) = min \begin{cases} a_u(x,y) & +s(x+1,y+1) \\ a_d(x,y) & +s(x+1,y-1) \end{cases}$$

Condición de frontera:

$$s(6,y) = B_y$$

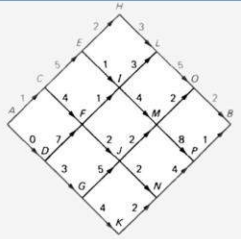
Óptimo:

$$s(0,y) = A_y$$

Solución 16

1.6 Un problema más complicado

Ahora complicaremos la situación para mostrar que para algunos problemas los argumentos para la función de valor puede contener más información que su simple ubicación (x,y).



Supongamos que despues de llegar a cualquier vértice en nuestro camino de A y B, cambiamos de dirección en vez de continuar en línea recta, si hacemos esto un costo adicional de 3 es añadido, no hay penalidad si continuamos en línea recta.

Para esto necesitamos saber en el vértice (x,y) tanto el costo mínimo como la penalidad al cambio de dirección.

Esto es un número asociado al vértice (x,y), en el caso de que escojamos ir arriba o abajo.

Función de valor óptimo

S(x,y,z) = El costo mínimo de la ruta que empieza en el vértice (x,y) a el punto B más la suma de las penalidades por cambio de dirección indicado por z, y donde z = 0 denota diagonalidad ascendente y z = 1, denota diagonalidad descendente.

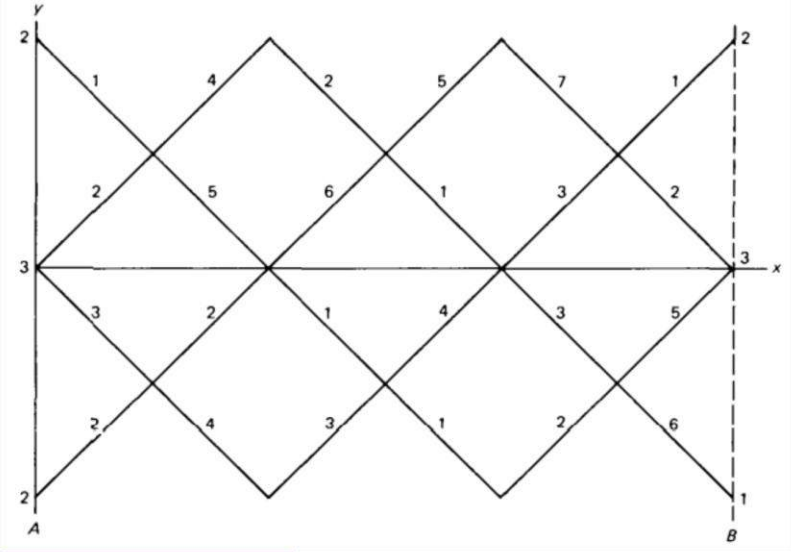
Nuestra relación de recurrencia queda como

$$S(x,y,z) = \begin{cases} min \begin{cases} a_u(x,y) & S(x+1,y+1,0) \\ a_d(x,y) & S(x+1,y+1,1)+3 \end{cases} & si \quad z=0 \\ min \begin{cases} a_u(x,y) & S(x+1,y+1,0)+3 \\ a_d(x,y) & S(x+1,y+1,1) \end{cases} & si \quad z=1 \end{cases}$$

Condición de frontera	Óptimo
$S(N,y,z) = 0$	$S(0,0,z) = 0$

Problema 1.6

Encuentra el valor de la ruta de costo mínimo desde la linea A hasta la B. los números mostrados a través de las lineas A y B son costos adicionales asociados con varios puntos iniciales y terminales.



Función de valor óptima:

Relación de recurrencia:

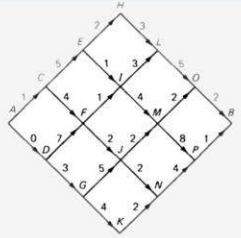
Condición de frontera:

Óptimo:

Solución

1.6 Un problema más complicado

Ahora complicaremos la situación para mostrar que para algunos problemas los argumentos para la función de valor puede contener más información que su simple ubicación (x,y).



Supongamos que despues de llegar a cualquier vértice en nuestro camino de A y B, cambiamos de dirección en vez de continuar en línea recta, si hacemos esto un costo adicional de 3 es añadido, no hay penalidad si continuamos en línea recta.

Para esto necesitamos saber en el vértice (x,y) tanto el costo mínimo como la penalidad al cambio de dirección.

Esto es un número asociado al vértice (x,y), en el caso de que escojamos ir arriba o abajo.

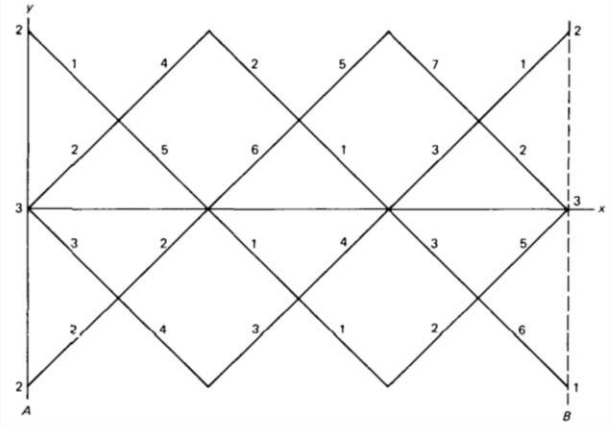
Función de valor óptimo

Nuestra relación de recurrencia queda como

Condición de frontera	Óptimo

Problema 1.7

Resuelve el siguiente problema sin los costos adicionales finales y la penalización de cambio de dirección



Función de valor óptima:

S(x,y,z) = El costo mínimo de la ruta que empieza en el vértice (x,y) a el punto (6,y) más la suma de las penalidades por cambio de dirección indicado por z, y donde z = 0 denota diagonalidad ascendente y z = 1, denota diagonalidad descendente.

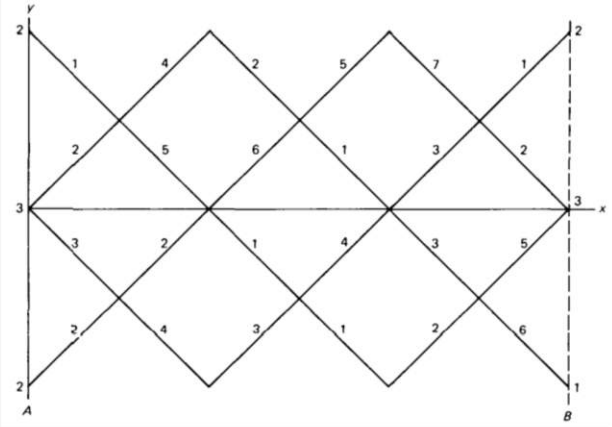
Relación de recurrencia:

$$S(x,y,z) = \begin{cases} min \begin{cases} a_u(x,y) & S(x+1,y+1,0) \\ a_d(x,y) & S(x+1,y+1,1)+3 \end{cases} & si \quad z=0 \\ min \begin{cases} a_u(x,y) & S(x+1,y+1,0)+3 \\ a_d(x,y) & S(x+1,y+1,1) \end{cases} & si \quad z=1 \end{cases}$$

Condición de frontera:	Óptimo:
$S(6,y,z)=0$	S(0,y,z) = 0

Problema 1.9

Da un procedimiento de programación dinámico para el problema de penalidad al cambiar de dirección.



Función de valor óptima:

S(x,y,z) = El costo mínimo de la ruta que empieza en el vértice (x,y) a el punto (0,y) más la suma de las penalidades por cambio de dirección indicado por z, y donde z = 0 denota diagonalidad ascendente y z = 1, denota diagonalidad descendente.

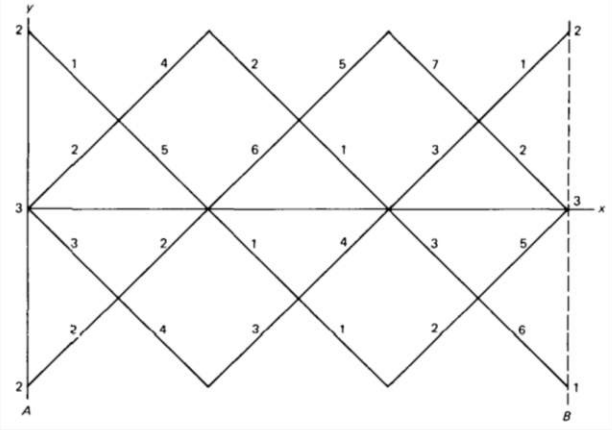
Relación de recurrencia:

$$S(x,y,z) = \begin{cases} min \begin{cases} a_u(x-1,y+1)+S(x-1,y+1,0) \\ a_d(x-1,y-1)+S(x-1,y-1,1)+3 \end{cases} & si \quad z=0 \\ min \begin{cases} a_u(x-1,y+1)+S(x-1,y+1,0)+3 \\ a_d(x-1,y-1)+S(x-1,y-1,1) \end{cases} & si \quad z=1 \end{cases}$$

Condición de frontera:	Óptimo:
$S(6,y,z)=0$	S(0,y,z) = 0

Problema 1.7

Resuelve el siguiente problema sin los costos adicionales finales y la penalización de cambio de dirección



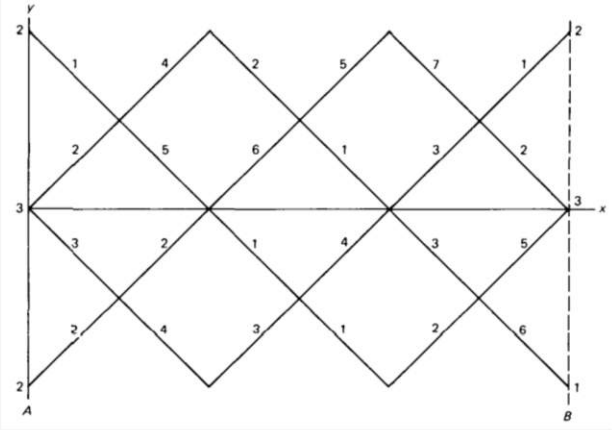
Función de valor óptima:

Relación de recurrencia:

Condición de frontera:	Óptimo:

Problema 1.9

Da un procedimiento de programación dinámico para el problema de penalidad al cambiar de dirección.



Función de valor óptima:

Relación de recurrencia:

Condición de frontera:	Óptimo:

1 Pregunta del consultor

Una vez que las ideas principales de programación dinámica son entendidas

1	*El uso de la función de valor óptimo.*
2	*La relación de recurrencia.*
3	*Y su recursiva solución trayendo correctamente la solución de problemas.*

El arte restante consiste en la elección correcta de los argumentos de la función de valor óptimo.

La pregunta del consultor

Supon que alguien esta tomando decisiones no óptimas por lo tanto decide pedir ayuda a un consultor, pero antes de que le diga algo le dice "Ahora tu manda y has las cosas óptimas de ahora en adelante".

La información mínima que recibe el, constituye los argumentos de la función de valor óptimo.

Pensando en la situación de esta manera y preguntando que información necesitas para pedir hacer soluciones óptimas, es llamada "la pregunta del consultor".

Etapa y estado

Hasta ahora hemos hablado de los argumentos de la función de valor óptimo para descubrir la situación.

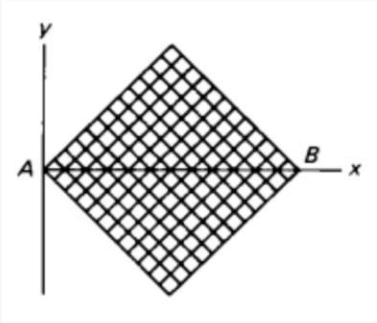
Generalmente es la variable x, la que nos indica cuantas decisiones hemos tomado hasta el momento.

Esa monotonia particular es llamada la variable de etapa o stage

Todas las demás variables son llamadas variables de estado o state.

Problema 1.11

El costo de una ruta es la sumatoria de los costos de sus arcos más el total del costo de cambio de dirección donde el cambio en la k-esima vez es k = 1,2,3,...



Función de valor óptima:
S(x,y,z,k) = el valor de la ruta óptima que conecta el vértice (x,y) con el vértice S(N,0), con dirección z y k veces cambio de dirección.

Relación de recurrencia:

$$S(x,y,1,k) = \begin{cases} a_u(x,y) + S(x+1,y+1,0,k+1) + k \\ a_d(x,y) + S(x+1,y-1,1,k) \end{cases}$$

$$S(x,y,0,k) = \begin{cases} a_u(x,y) + S(x+1,y+1,0,k) \\ a_d(x,y) + S(x+1,y-1,1,k+1) + k \end{cases}$$

Condición de frontera:	Óptimo:
S(N,0, z, k) = 0	S(0,0, z, 0)

1 Pregunta del consultor

Una vez que las ideas principales de programación dinámica son entendidas

1	*El uso de la función de valor óptimo.*
2	*La relación de recurrencia.*
3	*Y su recursiva solución trayendo correctamente la solución de problemas.*

El arte restante consiste en la elección correcta de los _____ de la función de valor óptimo.

La pregunta del consultor

Supon que alguien esta tomando decisiones no óptimas por lo tanto decide pedir ayuda a un consultor, pero antes de que le diga algo le dice "Ahora tu manda y has las cosas óptimas de ahora en adelante".

La información mínima que recibe el, constituye los argumentos de la función de valor óptimo.

Pensando en la situación de esta manera y preguntando que información necesitas para pedir hacer soluciones óptimas, es llamada "_____".

Etapa y estado

Hasta ahora hemos hablado de los argumentos de la función de valor óptimo para descubrir la situación.

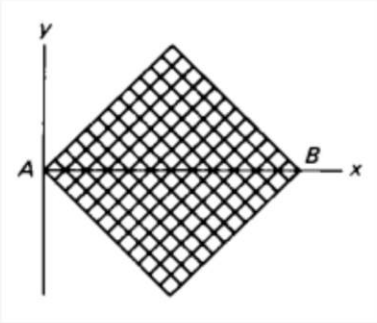
Generalmente es la variable x, la que nos indica cuantas decisiones hemos tomado hasta el momento.

Esa monotonia particular es llamada la variable de _____ o _____

Todas las demás variables son llamadas variables de _____ o _____.

Problema 1.11

El costo de una ruta es la sumatoria de los costos de sus arcos más el total del costo de cambio de dirección donde el cambio en la k-esima vez es k = 1,2,3,...



Función de valor óptima:

Relación de recurrencia:

Condición de frontera:	Óptimo:

Tesis

1 Conceptos Generales

La culiminación de los estudios profesionales es la obtención del titulo o grado.
Esto implica, elaborar una investigación cientifica en el áera.
Una vez terminada y aprobada por un comité de ivnestigación se debe presentar un examen en el que habra de defender la tesis.

1.1 ¿Qué es una tesis?

Significa proposición u opinión, proveien del latin *Thesis* y este del griego *Thésis*.

En estricto sentido es

La presentación formal de un trabajo academico, donde se explora un tema bajo un nuevo enfoque o demuestra una teoría propuesta.

Siguiendo

El rigor de un método cientifico de investigación

Toda tesis debe tener

1	Tema de investigación	5	Aspectos metodológicos
2	Problema de investigación	6	Análisis de datos
3	Marco teórico o estado del arte	7	Conclusiones
4	Hipotesis y objetivos	8	Bibliografia

1.1.1 Antecedentes

El primero en utlizarlo fue Platón en su obra La republica.

Sin embargo

Se atribuye al filoso aleman Georfe willhem Friedrich Hegel su uso sistematico, el cual nos legó el pensamiento dialéctico.

El cual consiste

Por una tesis, una anti tesis y una sintesis que conducen a la verdad, y debe seguir los siguientes pasos:

1 Planteamiento de la tesis

Es la proposición con la cual se quiere llegar a una verdad.

2 Antitesis

Busca negar la tesis y demostrar el inverso de esta.

3 Marco teórico o estado del arte

Mediante la sitensis se llega a la conclusión, ya sea que la tesis sea verdadera o falsa.

La definición cambia dependiendo el nivel de estudios

1 Tesis doctoral

Trabajo de investigación donde el candidato al doctorado analiza, propone y demuestra una nueva teoría y conocimientos ineditos.

2 Tesis Maestria

Trabajo de investigación donde se examina, analiza y comprueba o rechaza un teoría o conocimientos, con esto se busca reafirmar, rechazar o proponer un estudio en su disciplina.

3 Tesis de licenciatura

Trabajo de investigación donde se explora algún tema o teoría congruente a sus estudios, con rigor en el método de investigación, aunque solo se presentan recopilaciones con escasa profundidad, sin proponer conocimientos nuevos.

1 Conceptos Generales

La culiminación de los estudios profesionales es la obtención del titulo o grado.
Esto implica, elaborar una investigación cientifica en el áera.
Una vez terminada y aprobada por un comité de ivnestigación se debe presentar un examen en el que habra de defender la .

1.1 ¿Qué es una tesis?

Significa proposición u opinión, proveien del latin y este del griego .

En estricto sentido es

La presentación formal de un trabajo academico, donde se explora un tema bajo un nuevo enfoque o demuestra una teoría propuesta.

Siguiendo

El rigor de un método cientifico de investigación

Toda tesis debe tener

1	Tema de investigación	5	Aspectos metodológicos
2	Problema de investigación	6	Análisis de datos
3	Marco teórico o estado del arte	7	
4	Hipotesis y objetivos	8	

1.1.1 Antecedentes

El primero en utlizarlo fue en su obra La republica.

Sin embargo

Se atribuye al filoso aleman Georfe willhem Friedrich su uso sistematico, el cual nos legó el pensamiento dialéctico.

El cual consiste

Por una , una y una que conducen a la verdad, y debe seguir los siguientes pasos:

1 Planteamiento de la tesis

Es la proposición con la cual se quiere llegar a una verdad.

2 Antitesis

Busca negar la tesis y demostrar el inverso de esta.

3 Marco teórico o estado del arte

Mediante la sitensis se llega a la conclusión, ya sea que la tesis sea verdadera o falsa.

La definición cambia dependiendo el nivel de estudios

1 Tesis doctoral

Trabajo de investigación donde el candidato al doctorado analiza, propone y demuestra una nueva teoría y conocimientos ineditos.

2 Tesis Maestria

Trabajo de investigación donde se examina, analiza y comprueba o rechaza un teoría o conocimientos, con esto se busca reafirmar, rechazar o proponer un estudio en su disciplina.

3 Tesis de licenciatura

Trabajo de investigación donde se explora algún tema o teoría congruente a sus estudios, con rigor en el método de investigación, aunque solo se presentan recopilaciones con escasa profundidad, sin proponer conocimientos nuevos.

Manga
Microprocesadores

1 ¿What does the CPU do?

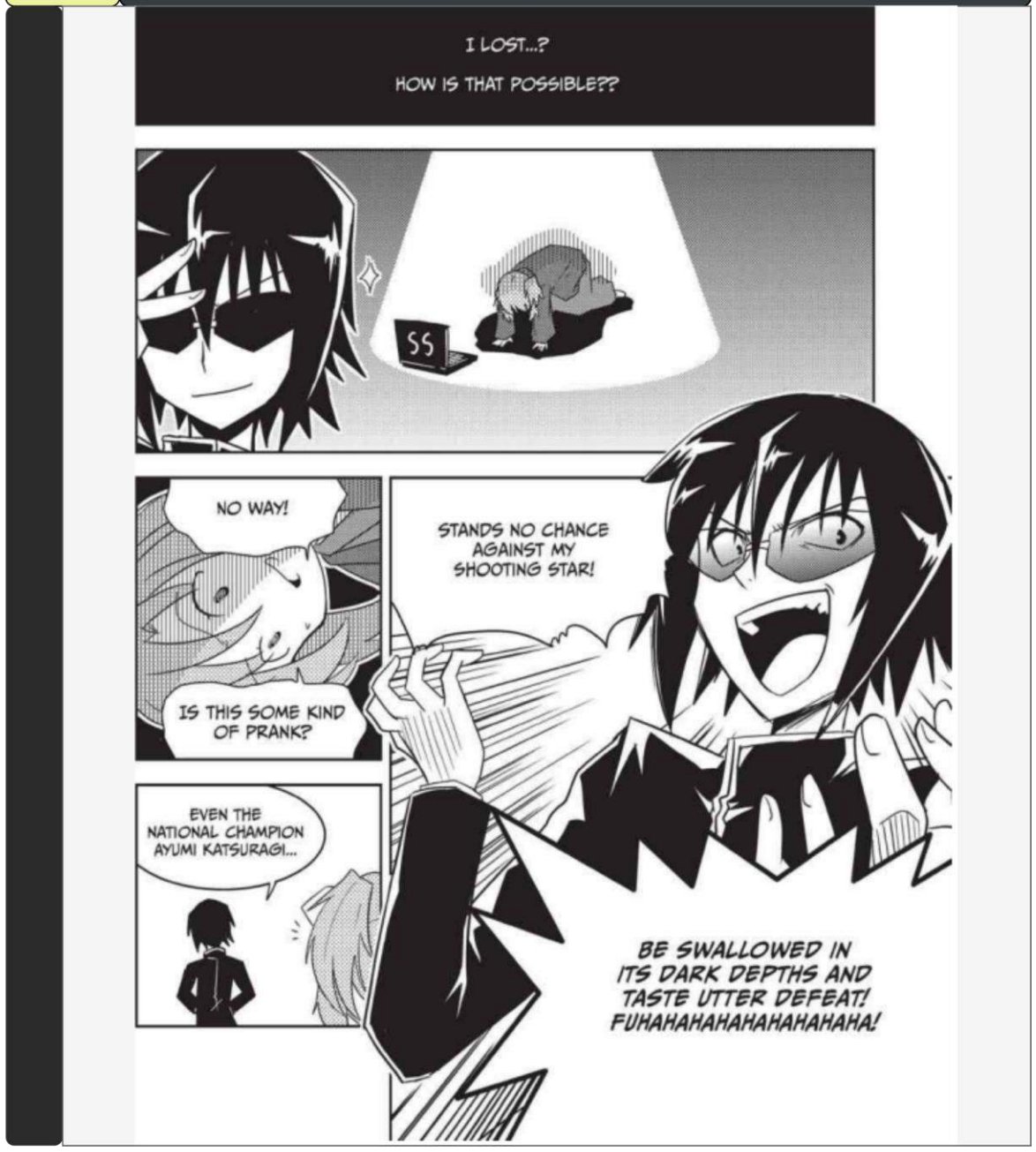
1 ¿What does the CPU do?

1 ¿What does the CPU do?

1 ¿What does the CPU do?

1 ¿What does the CPU do?

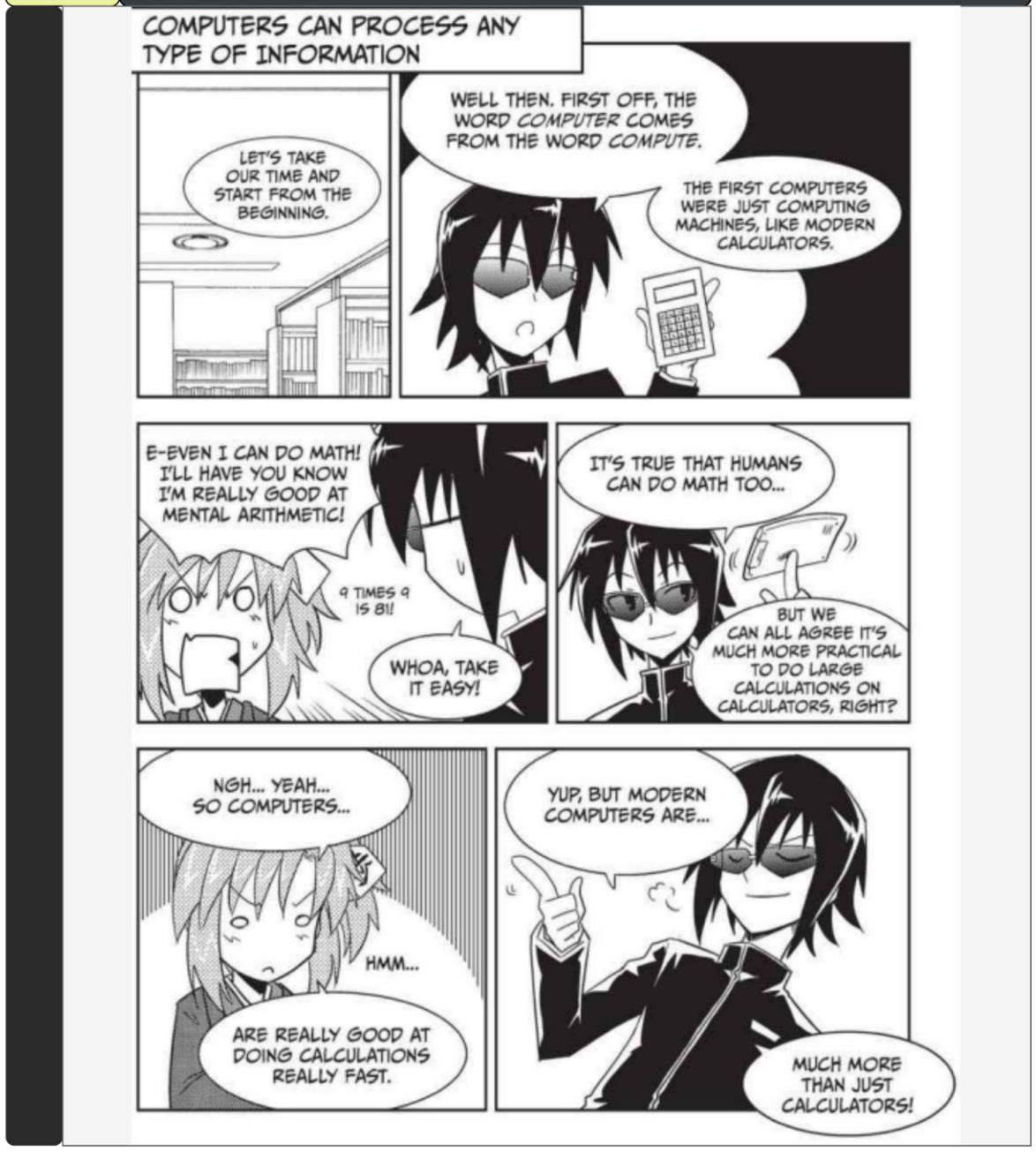
1 ¿What does the CPU do?



1 ¿What does the CPU do?

1 ¿What does the CPU do?

1 ¿What does the CPU do?



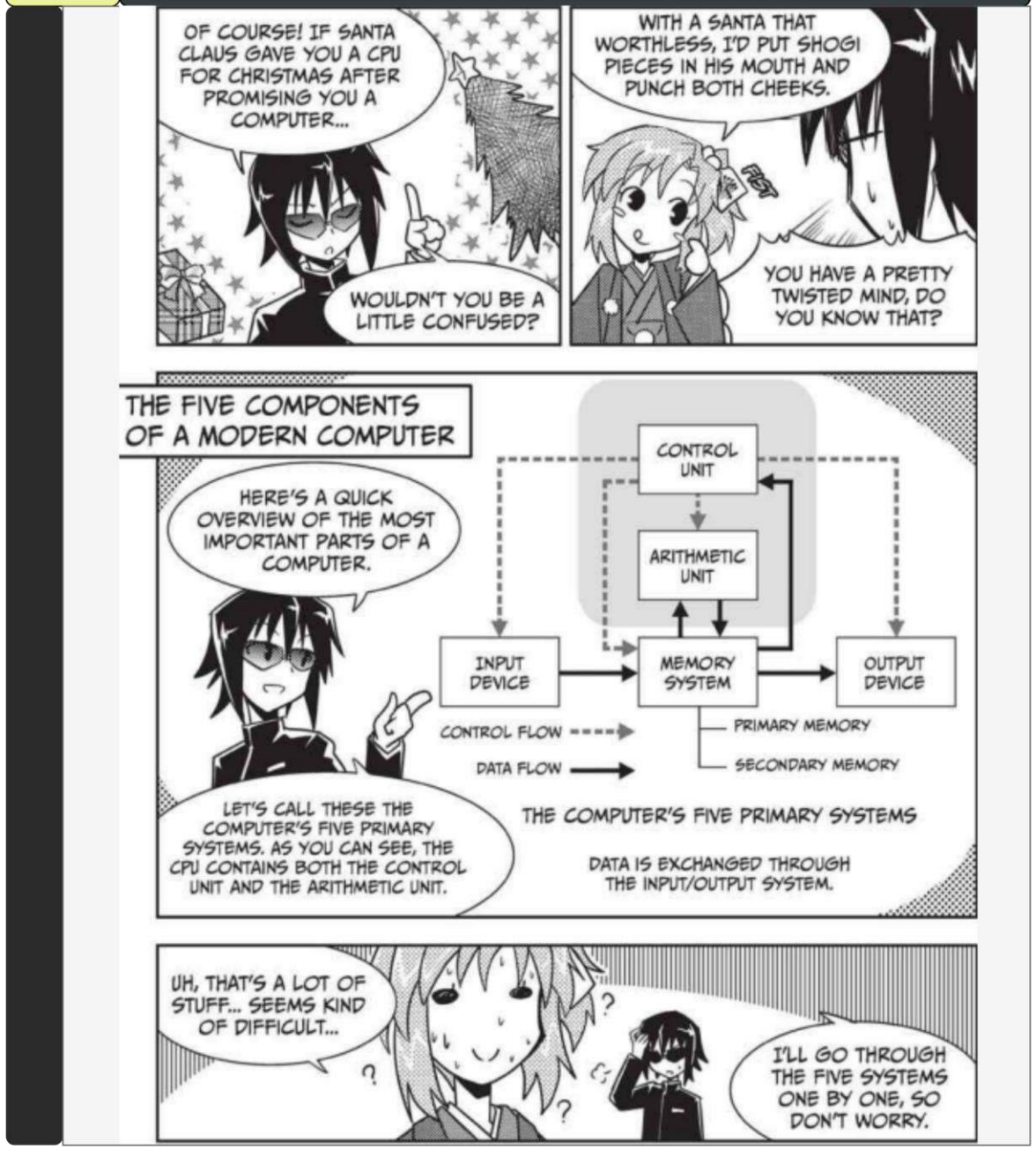
1

¿What does the CPU do?

1

¿What does the CPU do?

1 ¿What does the CPU do?



1 ¿What does the CPU do?

1 ¿Qué es lo que hace el CPU?

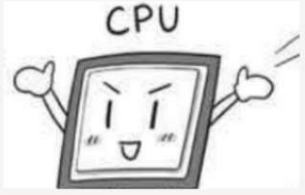
La palabra computadora viene de la palabra computar(calcular), las primeras computadoras solo eran computadoras que hacian calculos como las calculadoras modernas.

Información digital

Toda la información que es trasladada a 0s y 1s es llamada información digital.

El cerebro de la comptuadora

Es el nucleo de toda la tecnología digital, el CPU actúa como el cerebro.



CPU

El CPU es una abreviacion para decir:

Central processing unit o unidad de procesamiento central.

Operaciones del CPU

El CPU lleva a cabo dos tipos de operaciones las cuales son:

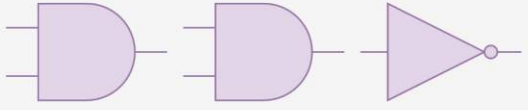
1 Operaciones aritméticas

Las únicas operaciones aritméticas que la computadora puede llevar a cabo es adición y sustracción

Mas Menos

2 Operaciones lógicas

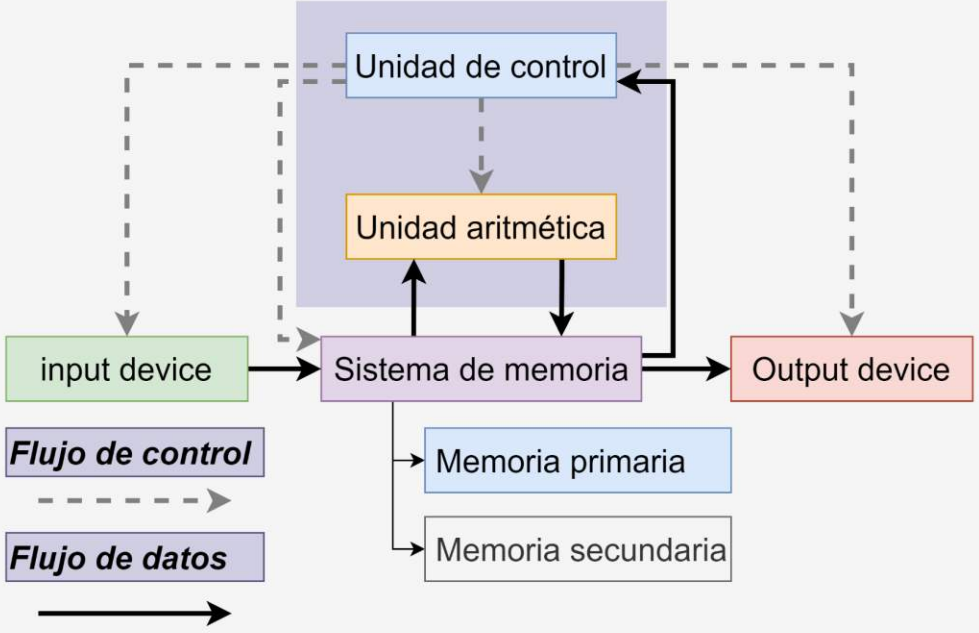
Operaciones logicas lidian con comparar pares de 1s y 0s



ADD OR NOT

1.1 Componentes de una computadora

A pesar de la importancia de el CPU, es necesario más componentes para que esta funcione, estas son llamadas los 5 sistemas primarios de las computadoras.



1.1.1 Dispositivos de entrada

Los dispositivos de entrada son sistemas que suplen a la computadora de instrucciones y datos

1.1.2 Dispositivos de salida

Los dispositivos de salida son sistemas que trasladan los datos internos en una presentación externa

1 ¿Qué es lo que hace el CPU?

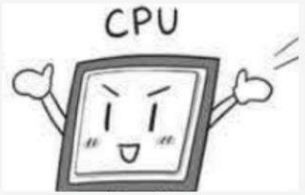
La palabra computadora viene de la palabra computar(calcular), las primeras computadoras solo eran computadoras que hacian calculos como las calculadoras modernas.

Información digital

Toda la información que es trasladada a 0s y 1s es llamada información _____.

El cerebro de la comptuadora

Es el nucleo de toda la tecnología digital, el CPU actúa como el cerebro.



CPU

El CPU es una abreviacion para decir:

_____ o _____.

Operaciones del CPU

El CPU lleva a cabo dos tipos de operaciones las cuales son:

1 Operaciones _____

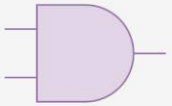
Las únicas operaciones aritméticas que la computadora puede llevar a cabo es adición y sustracción

Mas

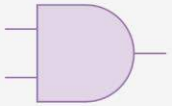
Menos

2 Operaciones _____

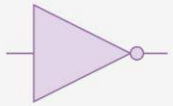
Operaciones logicas lidian con comparar pares de 1s y 0s



ADD



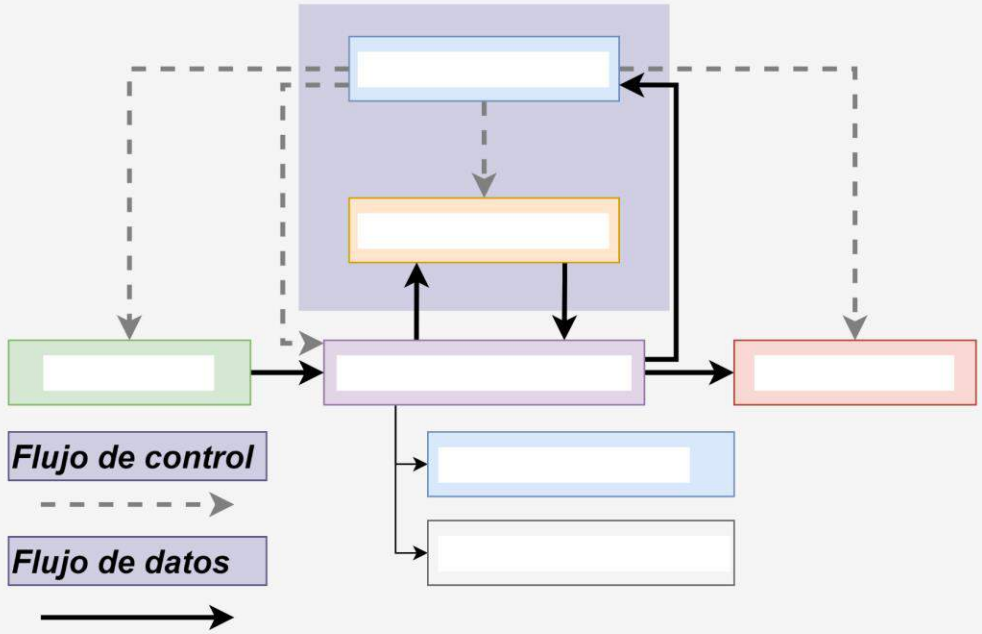
OR



NOT

1.1 Componentes de una computadora

A pesar de la importancia de el CPU, es necesario más componentes para que esta funcione, estas son llamadas los 5 sistemas primarios de las computadoras.



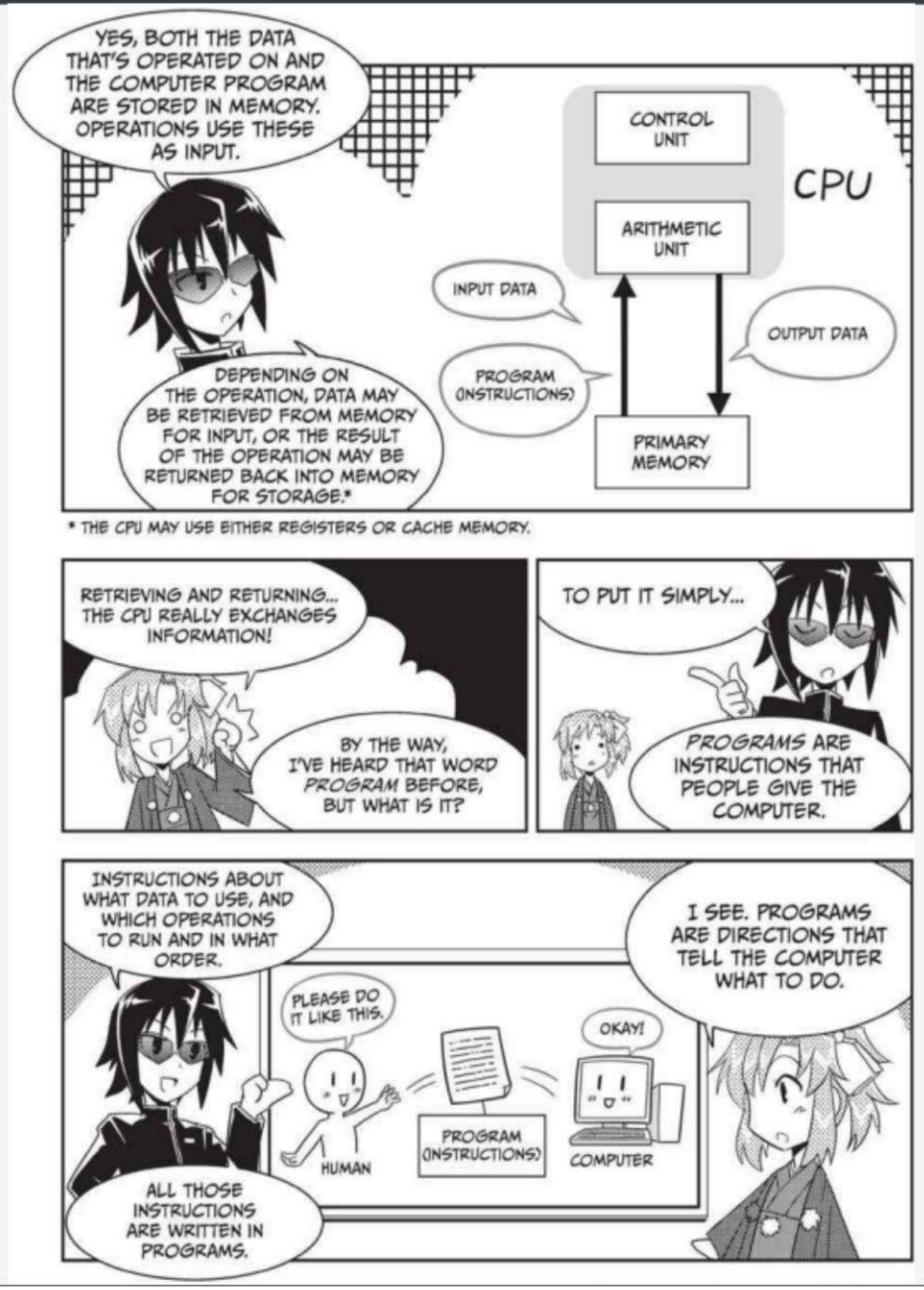
1.1.1 Dispositivos de entrada

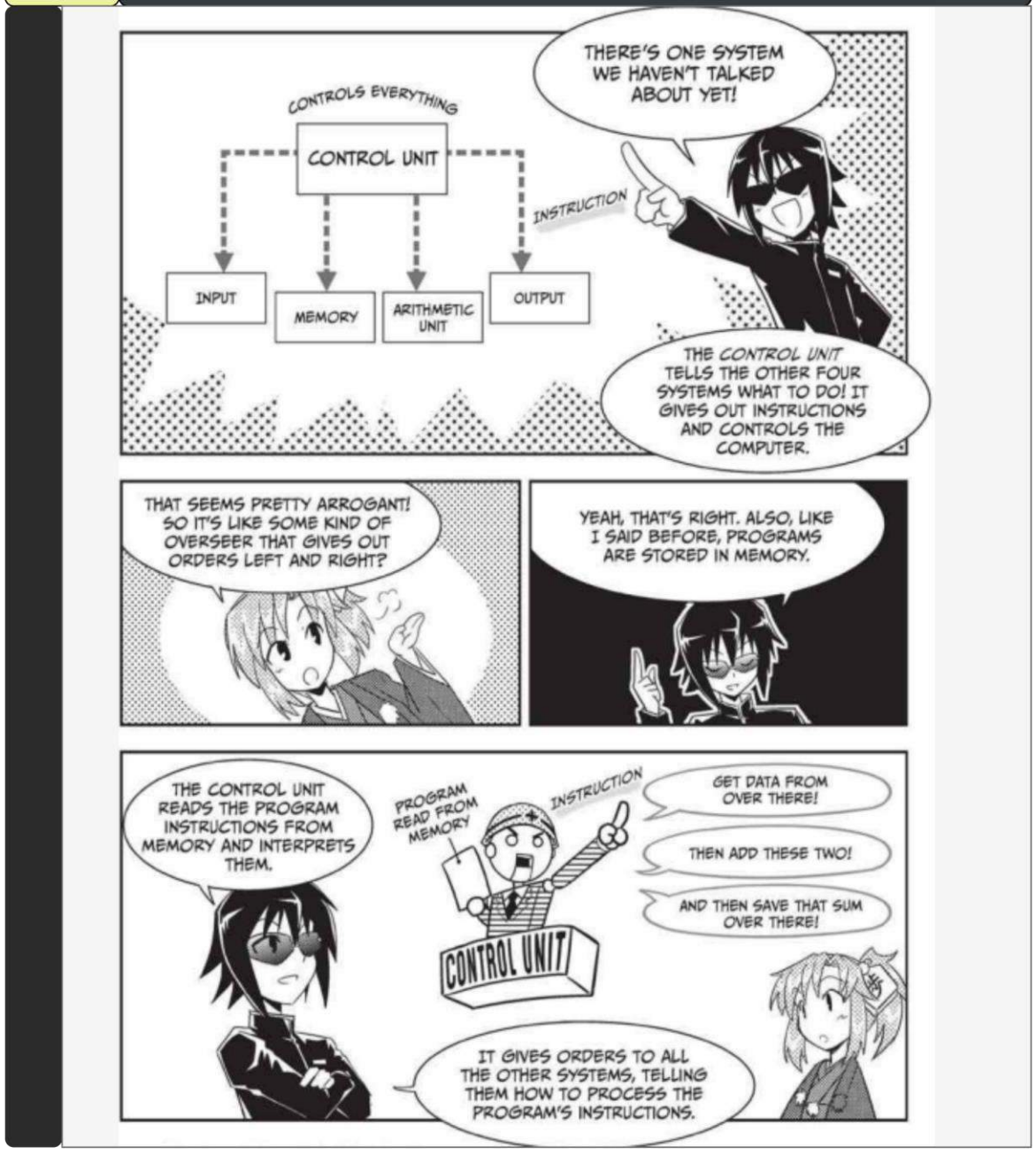
Los _____ son sistemas que suplen a la computadora de instrucciones y datos

1.1.2 Dispositivos de salida

Los _____ son sistemas que trasladan los datos internos en una presentación externa

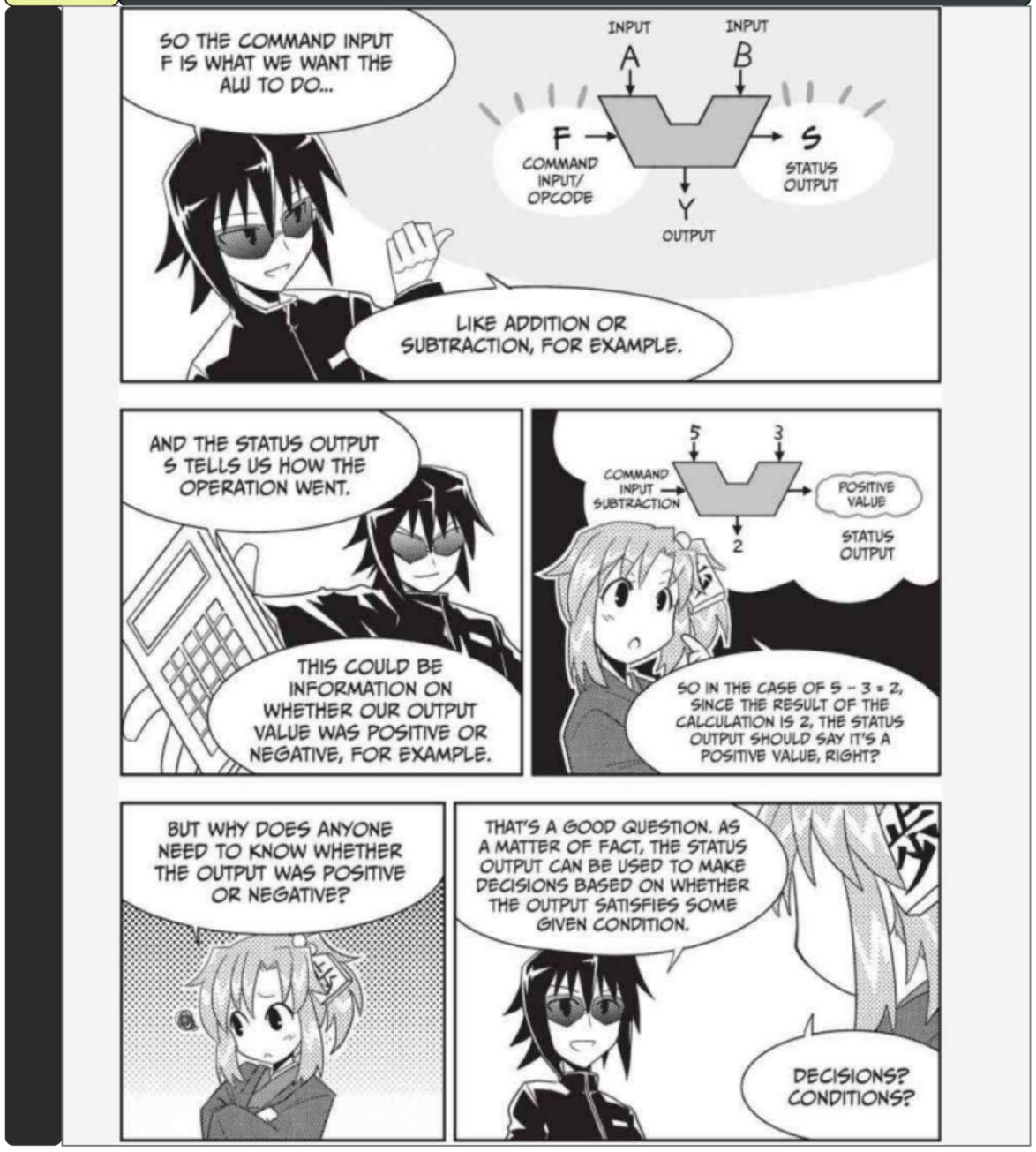
1 ¿What does the CPU do?



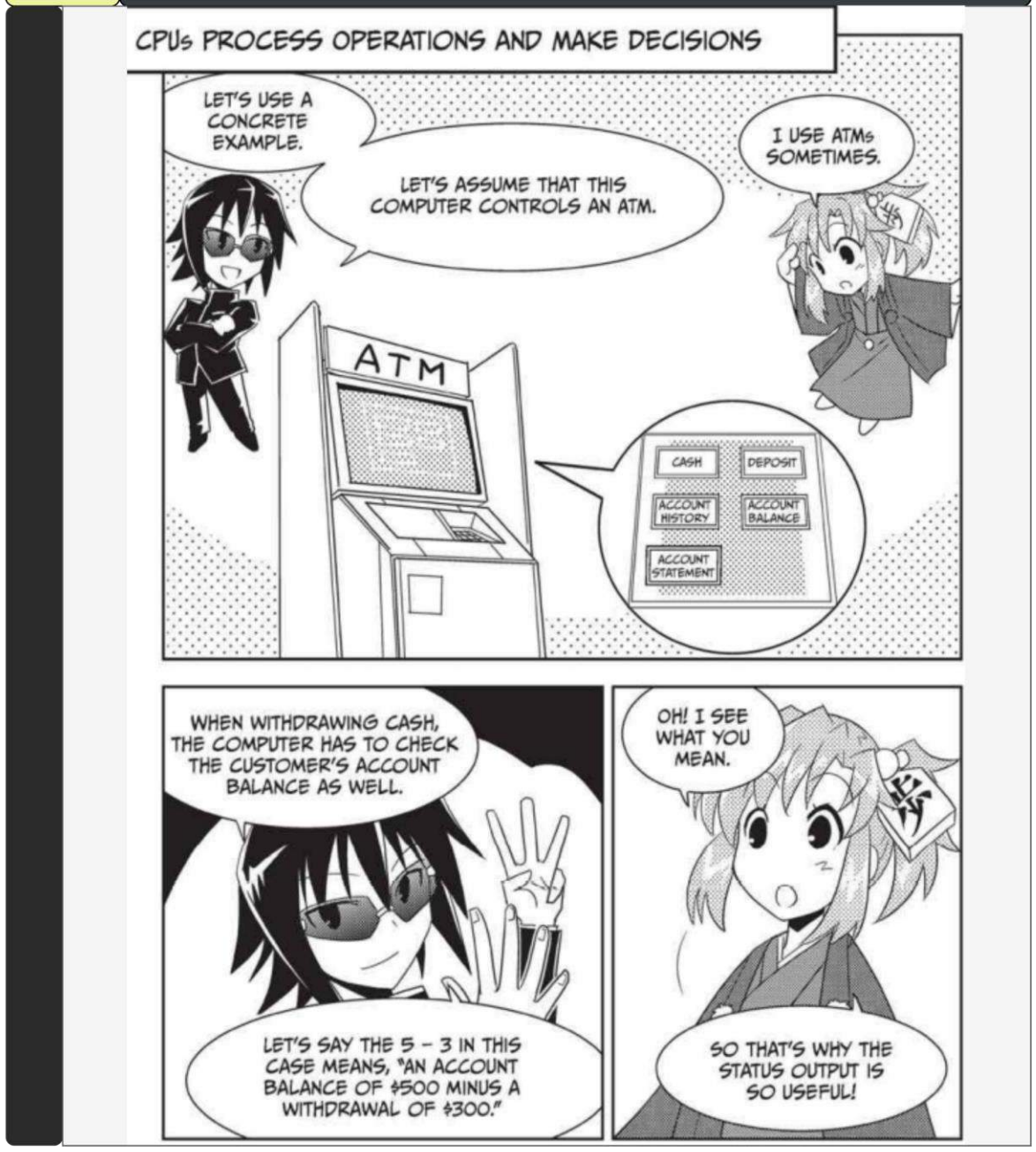


1 ¿What does the CPU do?

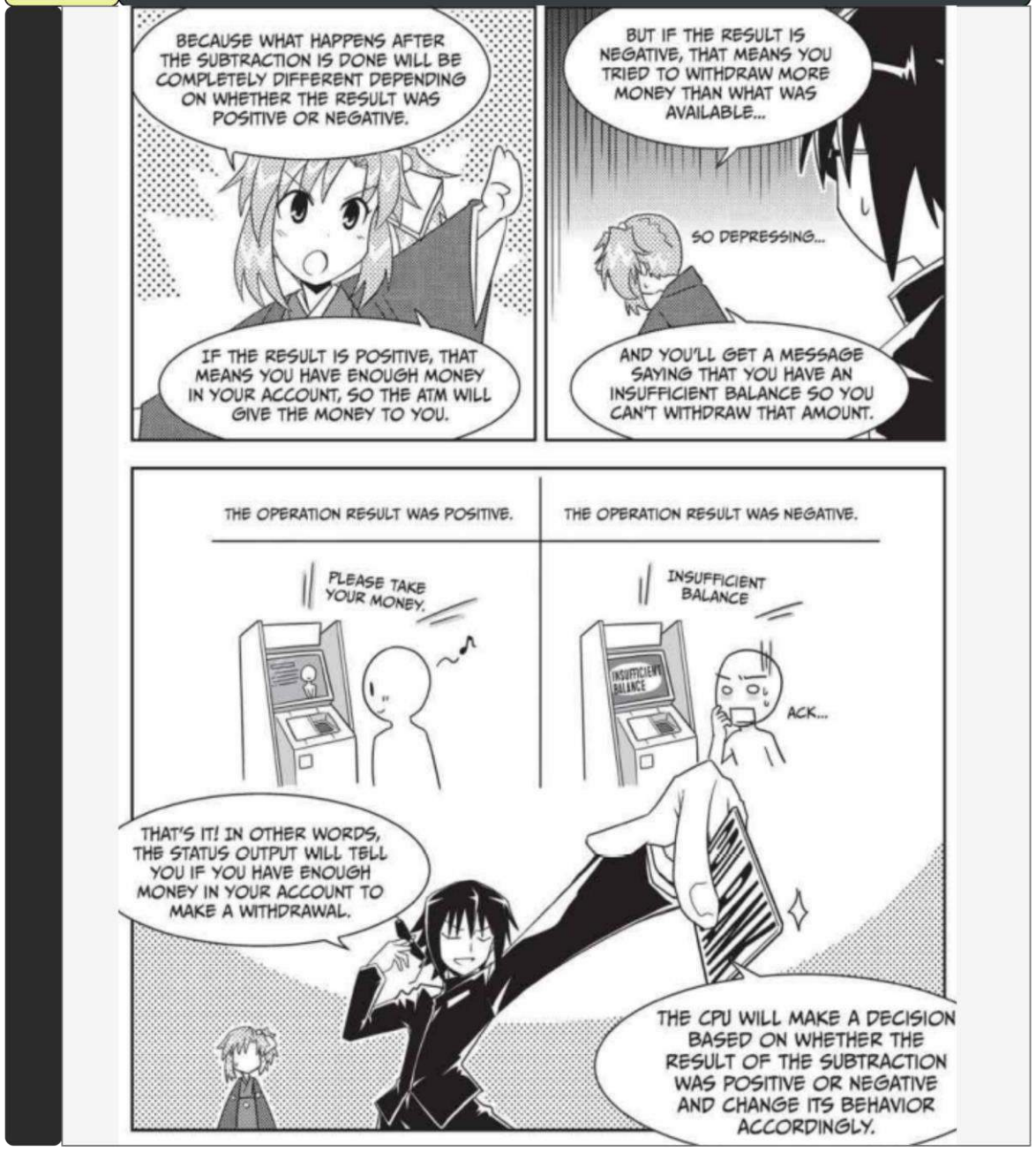
1 ¿What does the CPU do?



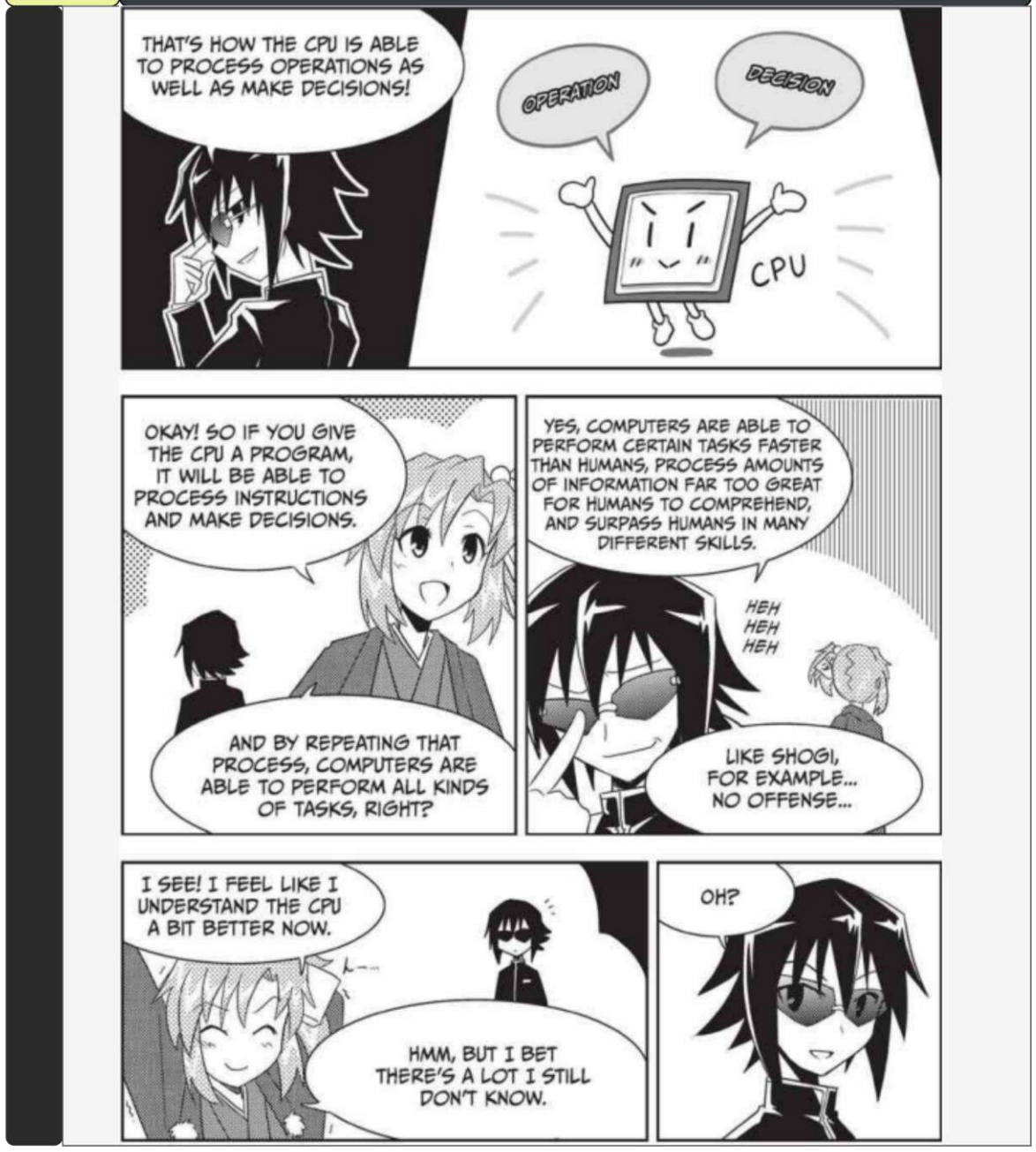
1 ¿What does the CPU do?



1 ¿What does the CPU do?

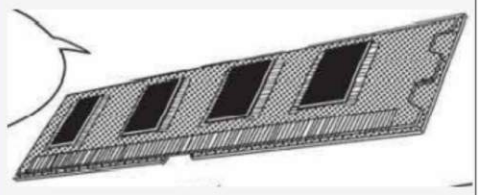


1 ¿What does the CPU do?



1.1.3 Sistema de memoria

El sistema de memoria es el responsable por guardar y obtener datos, la memoria viene en dos tipos primaria y secundaria. Mientras aprendemos del CPU nos enfocaremos en la memoria primaria.



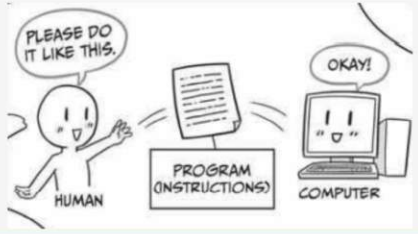
¿Por que es importante?

Esto es por que cuando la CPU realiza operaciones, esta siempre necesita operar con algun tipo de dato guardado en memoria.

Tanto los datos que son calculados como el programa son guardados en memoria, las operaciones usan estos datos como entrada.

¿Qué es un programa?

Los programas son instrucciones que las personas dan a la computadora, que datos usar y que operaciones usar.



1.1.4 Unidad de control

La unidad de control le dice a los otros 4 sistemas que hacer, da instrucciones y controla computadora.

¿Como lo hace?

La unidad de control lee las instrucciones de un programa desde la memoria y los interpreta, dando ordenes a los otros sistemas sobre como procesar las instrucciones del programa.

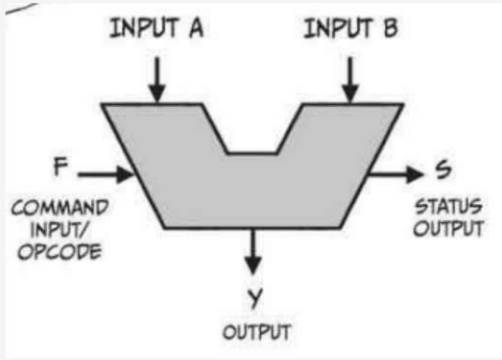
1.1.5 Unidad aritmetica (ALU)

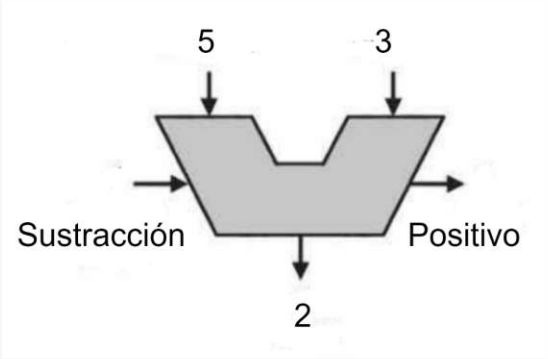
Alu es lo que realiza las operaciones dentro de un CPU, significa Airthmeti logic unit

¿Como se ve?

Un arithmetic logic unit se mira así:

Ejemplo 5-3 = 2

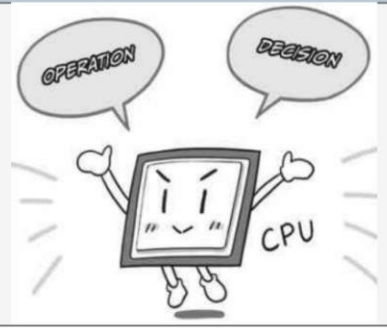




Input e input B son los datos que necesita para producir el output y, el command input es la operación que deseamos que haga, y por último el estatus output nos dice si el resultado fue positivo o negativo.

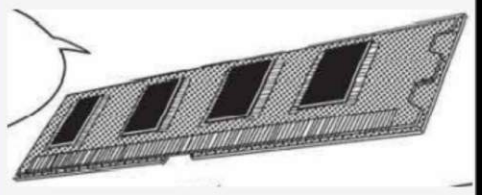
El estado de salida puede ayudarnos a tomar la decision correcta por ejemplo si en el banco retiras más dinero del que tienes, este regresara negativo.

Esta es la memoria en que una CPU es capaz de procesar operaciones así como decisiones.



1.1.3 Sistema de memoria

El sistema de memoria es el responsable por guardar y obtener datos, la memoria viene en dos tipos primaria y secundaria. Mientras aprendemos del CPU nos enfocaremos en la memoria primaria.



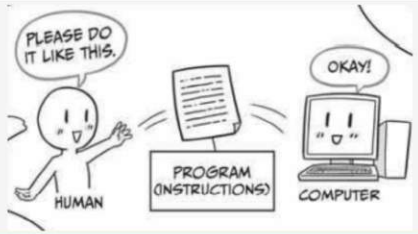
¿Por que es importante?

Esto es por que cuando la CPU realiza operaciones, esta siempre necesita operar con algun tipo de dato guardado en memoria.

Tanto los datos que son calculados como el programa son guardados en memoria, las operaciones usan estos datos como entrada.

¿Qué es un programa?

Los programas son instrucciones que las personas dan a la computadora, que datos usar y que operaciones usar.



1.1.4 Unidad de control

La unidad de control le dice a los otros 4 sistemas que hacer, da instrucciones y controla computadora.

¿Como lo hace?

La unidad de control lee las instrucciones de un programa desde la memoria y los interpreta, dando ordenes a los otros sistemas sobre como procesar las instrucciones del programa.

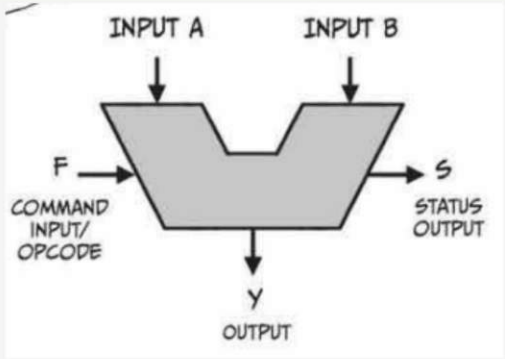
1.1.5 Unidad aritmetica (ALU)

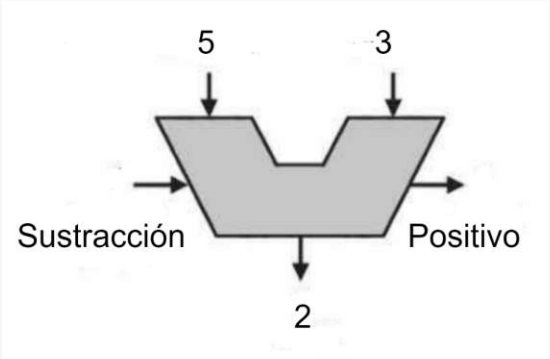
Alu es lo que realiza las operaciones dentro de un CPU, significa Airthmeti logic unit

¿Como se ve?

Un arithmetic logic unit se mira así:

Ejemplo 5-3 = 2

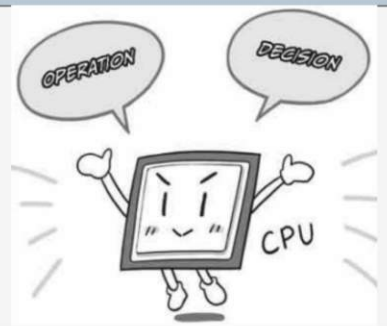


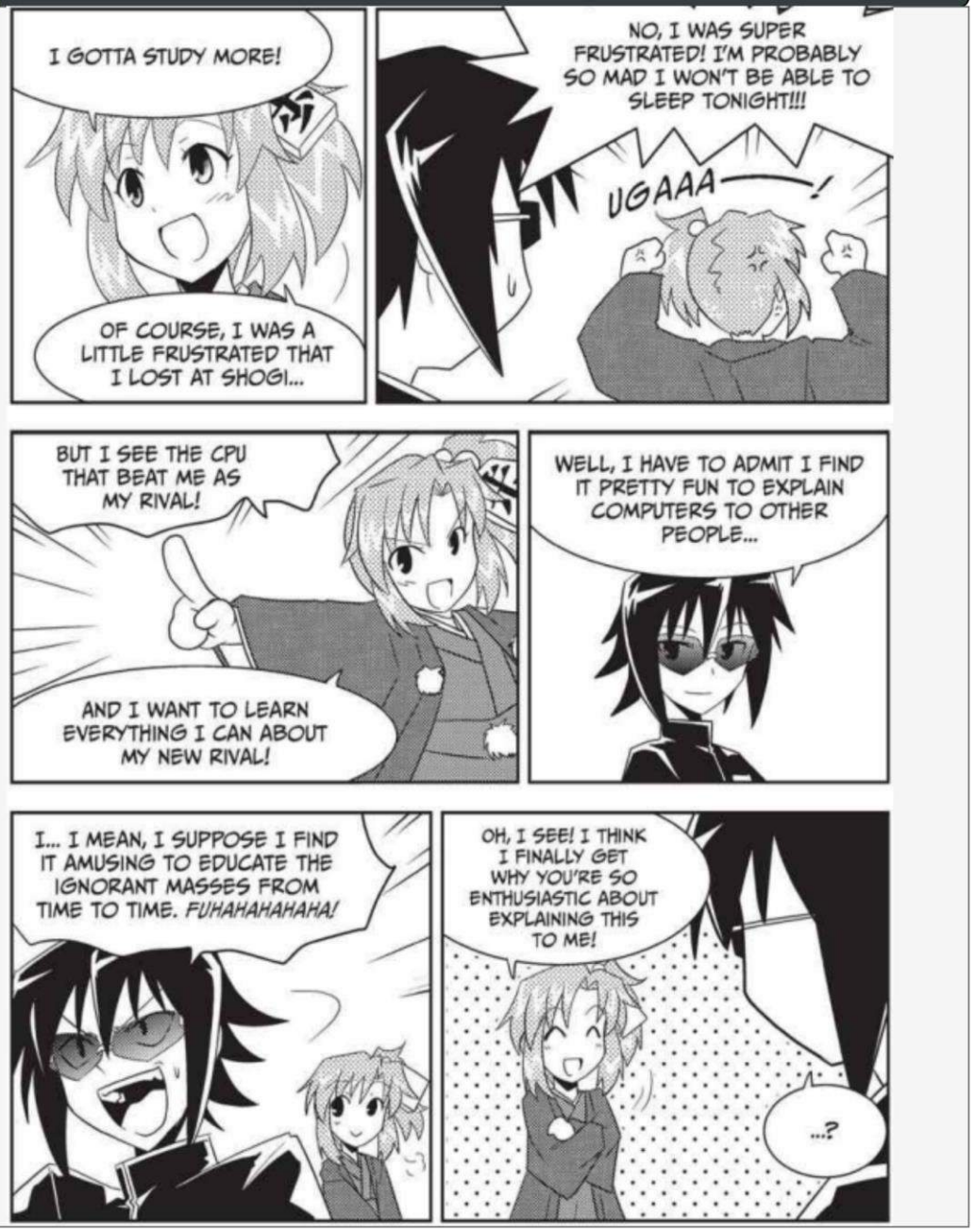


Input e input B son los datos que necesita para producir el output y, el command input es la operación que deseamos que haga, y por último el estatus output nos dice si el resultado fue positivo o negativo.

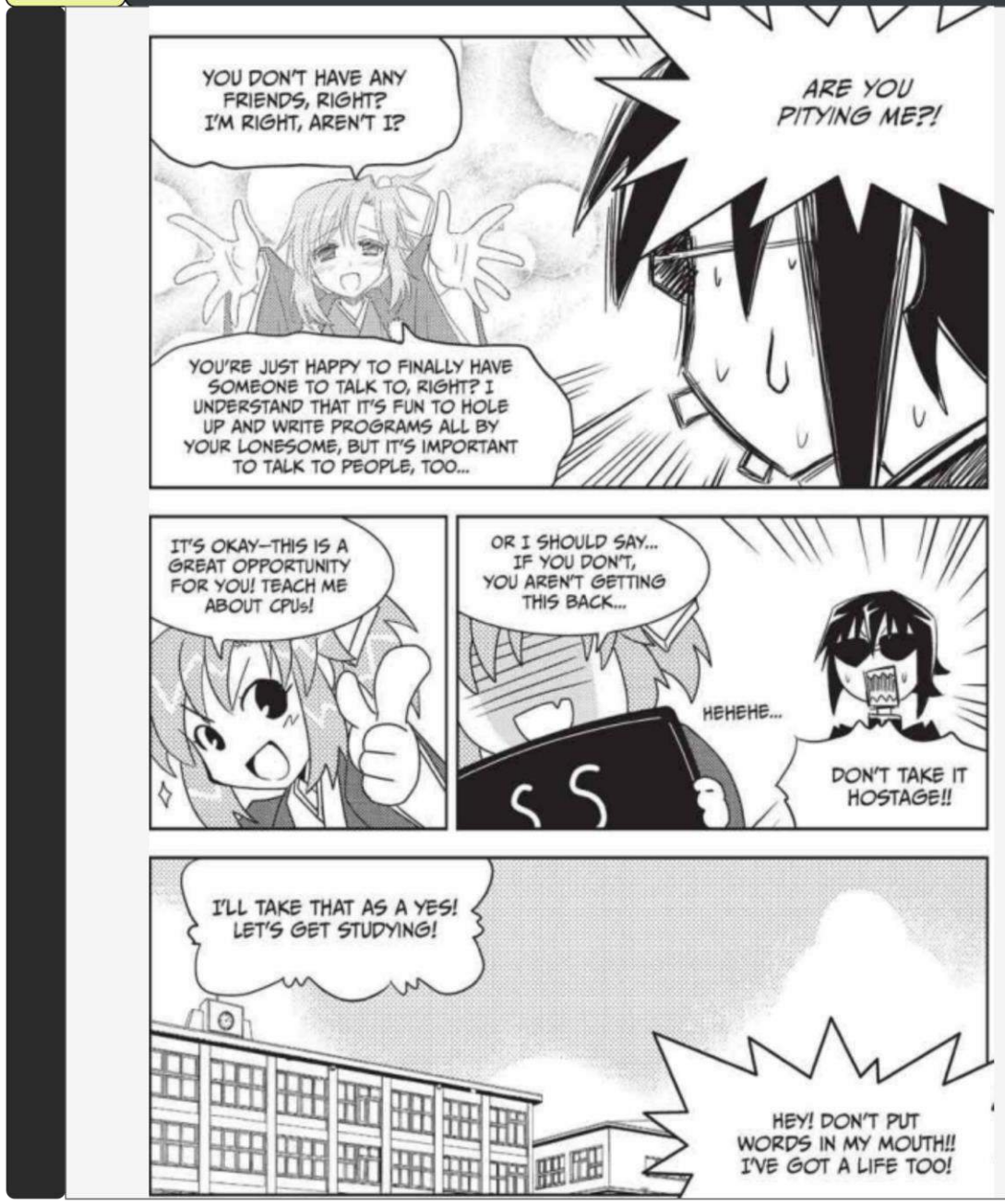
El estado de salida puede ayudarnos a tomar la decision correcta por ejemplo si en el banco retiras más dinero del que tienes, este regresara negativo.

Esta es la memoria en que una CPU es capaz de procesar operaciones así como decisiones.





1 ¿What does the CPU do?



Otros

1 Ciclo de Otto

1.1 ¿Qué es?

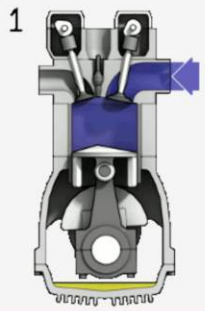
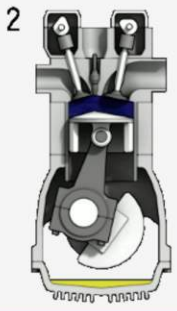
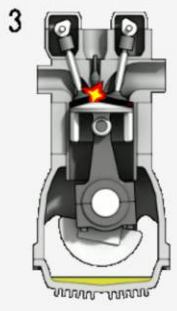
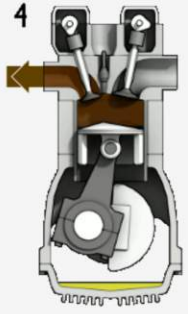
Es el ciclo termodinámico que se aplica a motores de combustión interna de ecendido provocado por una chispa electrica.

Inventor

Inventado por Nicolaus Otto en 1876.

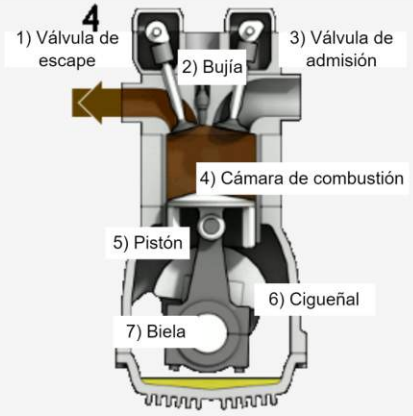
1.2 Ciclo de cuatro tiempos

Es el más utlizado en automoviles ya que tiene buen rendimiento y contamina mucho menos que el de dos tiempos.

1) Admisión	2) Compresión	3) Expansión	4) Escape
			

Partes del Pistón

1	Válvula de escape
2	Bujía
3	Válvula de admisión
4	Camara de combustión
5	Pistón
6	Cigueñal
7	Biela



2 Gasolina Magna o Premium

Necesitamos saber que tipo de gasolina necesita nuestro auto, lo cual es vital para la vida del motor, y su rendimiento, para eso es muy importante saber que es el octanaje.

1.2 ¿Qué es Octanaje?

Son hidrocarburos quedeterminan la forma en que la gasolina hace combustíon, a mayor octanaje la gasolina tendra mayor capacidad antidetonante es decir, tendra mayor capacidad de compresión en el segundo ciclo de Otto.

1.3 Octanaje de Magna y Premium

Gasolina Premium

92 octanos, mayor capacidad de compresión antes de explotar, además de ser más caro por su proceso de refinación tardado y sin azufre..

Gasolina Magna

87 Octanos, menor capacidad de compresión.

1.4 ¿Que tipo de gasolina necesito?

Usualmente viene en el manual del propietario, pero si no este puede verse en la tapa de combustible, se vera la layenda AKY o RON y el número de octanos.

RON 91 o AKY 87	Requiere gasolina Magna
RON 95 o AKY 91	Requiere gasolina Premium

1.5 Consecuencias de usar una gasolina incorrecta o mezclarlos

En caso de poner gasolina magna a un carro que requiere premium, este comenzara a explotar cuando el pistón sube, sin que haya hecho el recorrido total, ocasionaria forzar los pistones, bielas y el cigueñal, haciendo que se aflojen, provocando un cascabeleo y perdida de potencia en el motor y reducción de su vida util.

1 Ciclo de Otto

1.1 ¿Qué es?

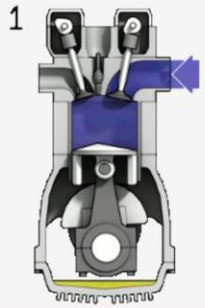
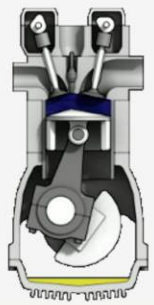
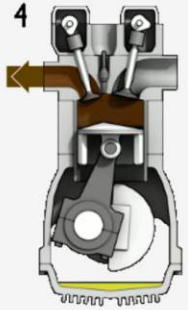
Es el ciclo termodinámico que se aplica a motores de combustión interna de ecendido provocado por una _____ electrica.

Inventor

Inventado por _____ en 1876.

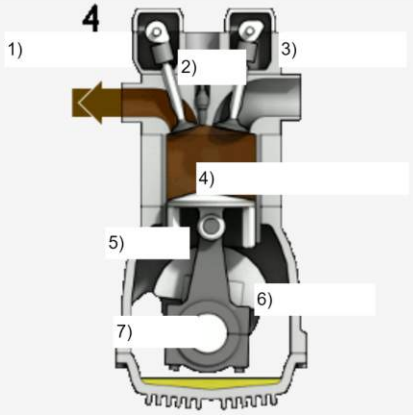
1.2 Ciclo de cuatro tiempos

Es el más utlizado en automoviles ya que tiene buen rendimiento y contamina mucho menos que el de dos tiempos.

1) _____	2) _____	3) _____	4) _____
			

Partes del Pistón

1	Válvula de escape
2	Bujía
3	Válvula de admisión
4	Camara de combustión
5	Pistón
6	Cigueñal
7	Biela



2 Gasolina Magna o Premium

Necesitamos saber que tipo de gasolina necesita nuestro auto, lo cual es vital para la vida del motor, y su rendimiento, para eso es muy importante saber que es el octanaje.

1.2 ¿Qué es Octanaje?

Son hidrocarburos quedeterminan la forma en que la gasolina hace _____, a mayor octanaje la gasolina tendra mayor capacidad antidetonante es decir, tendra mayor capacidad de compresión en el segundo ciclo de Otto.

1.3 Octanaje de Magna y Premium

Gasolina Premium

____ octanos, mayor capacidad de compresión antes de explotar, además de ser más caro por su proceso de refinación tardado y sin azufre..

Gasolina Magna

____ Octanos, menor capacidad de compresión.

1.4 ¿Que tipo de gasolina necesito?

Usualmente viene en el manual del propietario, pero si no este puede verse en la tapa de combustible, se vera la layenda ____ o ____ y el número de octanos.

RON ____ o AKY ____	Requiere gasolina Magna
RON ____ o AKY ____	Requiere gasolina Premium

1.5 Consecuencias de usar una gasolina incorrecta o mezclarlos

En caso de poner gasolina magna a un carro que requiere premium, este comenzara a explotar cuando el pistón sube, sin que haya hecho el recorrido total, ocasionaria forzar los pistones, bielas y el cigueñal, haciendo que se aflojen, provocando un cascabeleo y perdida de potencia en el motor y reducción de su vida util.

Estados de México

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?	32

Michoacán	Morelia
Nuevo León	Monterrey
Veracruz	Xalapa
Baja California	Mexicali
San Luis Potosí	San Luis Potosí
Puebla	Puebla
Campeche	Campeche
Aguascalientes	Aguascalientes
Chiapas	Tuxtla gutiérrez
Querétaro	Queretaro
Yucatán	Mérida
Chihuahua	Chihuahua
Guanajuato	Guanajuato
Sinaloa	Culiacan
Hidalgo	Pachuca
Ciudad de México	Ciudad de México
Zacatecas	Zacatecas
Quintana Roo	Chetumal
Morelos	Cuernavaca
Oaxaca	Oaxaca

Jalisco	Guadalajara
Nayarit	Tepic
Coahuila	Saltillo
Estado de méxico	Toluca
Colima	Colima
Durango	Durango
Baja California Sur	La Paz
Guerrero	Chilpancingo

Sonora	Hermosillo
Tabasco	Villahermosa
Tamaulipas	Ciudad victoria
Tlaxcala	Tlaxcala

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

1

Estados de México

Escribe los estados y capitales

¿Cuantos estados tiene México?

Michoacán	
Nuevo León	
Veracruz	
Baja California	
San Luis Potosí	
Puebla	
Campeche	
Aguascalientes	
Chiapas	
Querétaro	
Yucatán	
Chihuahua	
Guanajuato	
Sinaloa	
Hidalgo	
Ciudad de México	
Zacatecas	
Quintana Roo	
Morelos	
Oaxaca	

Jalisco	
Nayarit	
Coahuila	
Estado de méxico	
Colima	
Durango	
Baja California Sur	
Guerrero	

Sonora	
Tabasco	
Tamaulipas	
Tlaxcala	

Estados de México

Sistema solar

1

Sistema solar

MERCURIO | VENUS | TIERRA | MARTE | JÚPITER | SATURNO | URANO | NEPTUNO | PLUTÓN

1

Sistema solar

1

Sistema solar

1

Sistema solar

1

Sistema solar

1

Sistema solar

1

Sistema solar

Tablas multiplicar

* 1	
1 * 1 =	1
1 * 2 =	2
1 * 3 =	3
1 * 4 =	4
1 * 5 =	5
1 * 6 =	6
1 * 7 =	7
1 * 8 =	8
1 * 9 =	9
1 * 10 =	10

* 2	
2 * 1 =	2
2 * 2 =	4
2 * 3 =	6
2 * 4 =	8
2 * 5 =	10
2 * 6 =	12
2 * 7 =	14
2 * 8 =	16
2 * 9 =	18
2 * 10 =	20

* 3	
3 * 1 =	3
3 * 2 =	6
3 * 3 =	9
3 * 4 =	12
3 * 5 =	15
3 * 6 =	18
3 * 7 =	21
3 * 8 =	24
3 * 9 =	27
3 * 10 =	30

* 4	
4 * 1 =	4
4 * 2 =	8
4 * 3 =	12
4 * 4 =	16
4 * 5 =	20
4 * 6 =	24
4 * 7 =	28
4 * 8 =	32
4 * 9 =	36
4 * 10 =	40

* 5	
5 * 1 =	5
5 * 2 =	10
5 * 3 =	15
5 * 4 =	20
5 * 5 =	25
5 * 6 =	30
5 * 7 =	35
5 * 8 =	40
5 * 9 =	45
5 * 10 =	50

* 6	
6 * 1 =	6
6 * 2 =	12
6 * 3 =	18
6 * 4 =	24
6 * 5 =	30
6 * 6 =	36
6 * 7 =	42
6 * 8 =	48
6 * 9 =	54
6 * 10 =	60

* 7	
7 * 1 =	7
7 * 2 =	14
7 * 3 =	21
7 * 4 =	28
7 * 5 =	35
7 * 6 =	42
7 * 7 =	49
7 * 8 =	56
7 * 9 =	63
7 * 10 =	70

* 8	
8 * 1 =	8
8 * 2 =	16
8 * 3 =	24
8 * 4 =	32
8 * 5 =	40
8 * 6 =	48
8 * 7 =	56
8 * 8 =	64
8 * 9 =	72
8 * 10 =	80

* 9	
9 * 1 =	9
9 * 2 =	18
9 * 3 =	27
9 * 4 =	36
9 * 5 =	45
9 * 6 =	54
9 * 7 =	63
9 * 8 =	72
9 * 9 =	81
9 * 10 =	90

* 10	
10 * 1 =	10
10 * 2 =	20
10 * 3 =	30
10 * 4 =	40
10 * 5 =	50
10 * 6 =	60
10 * 7 =	70
10 * 8 =	80
10 * 9 =	90
10 * 10 =	100

* 1	
1 * 1 =	
1 * 2 =	
1 * 3 =	
1 * 4 =	
1 * 5 =	
1 * 6 =	
1 * 7 =	
1 * 8 =	
1 * 9 =	
1 * 10 =	

* 2	
2 * 1 =	
2 * 2 =	
2 * 3 =	
2 * 4 =	
2 * 5 =	
2 * 6 =	
2 * 7 =	
2 * 8 =	
2 * 9 =	
2 * 10 =	

* 3	
3 * 1 =	
3 * 2 =	
3 * 3 =	
3 * 4 =	
3 * 5 =	
3 * 6 =	
3 * 7 =	
3 * 8 =	
3 * 9 =	
3 * 10 =	

* 4	
4 * 1 =	
4 * 2 =	
4 * 3 =	
4 * 4 =	
4 * 5 =	
4 * 6 =	
4 * 7 =	
4 * 8 =	
4 * 9 =	
4 * 10 =	

* 5	
5 * 1 =	
5 * 2 =	
5 * 3 =	
5 * 4 =	
5 * 5 =	
5 * 6 =	
5 * 7 =	
5 * 8 =	
5 * 9 =	
5 * 10 =	

* 6	
6 * 1 =	
6 * 2 =	
6 * 3 =	
6 * 4 =	
6 * 5 =	
6 * 6 =	
6 * 7 =	
6 * 8 =	
6 * 9 =	
6 * 10 =	

* 7	
7 * 1 =	
7 * 2 =	
7 * 3 =	
7 * 4 =	
7 * 5 =	
7 * 6 =	
7 * 7 =	
7 * 8 =	
7 * 9 =	
7 * 10 =	

* 8	
8 * 1 =	
8 * 2 =	
8 * 3 =	
8 * 4 =	
8 * 5 =	
8 * 6 =	
8 * 7 =	
8 * 8 =	
8 * 9 =	
8 * 10 =	

* 9	
9 * 1 =	
9 * 2 =	
9 * 3 =	
9 * 4 =	
9 * 5 =	
9 * 6 =	
9 * 7 =	
9 * 8 =	
9 * 9 =	
9 * 10 =	

* 10	
10 * 1 =	
10 * 2 =	
10 * 3 =	
10 * 4 =	
10 * 5 =	
10 * 6 =	
10 * 7 =	
10 * 8 =	
10 * 9 =	
10 * 10 =	

* 1	
1 * 1 =	
1 * 2 =	
1 * 3 =	
1 * 4 =	
1 * 5 =	
1 * 6 =	
1 * 7 =	
1 * 8 =	
1 * 9 =	
1 * 10 =	

* 2	
2 * 1 =	
2 * 2 =	
2 * 3 =	
2 * 4 =	
2 * 5 =	
2 * 6 =	
2 * 7 =	
2 * 8 =	
2 * 9 =	
2 * 10 =	

* 3	
3 * 1 =	
3 * 2 =	
3 * 3 =	
3 * 4 =	
3 * 5 =	
3 * 6 =	
3 * 7 =	
3 * 8 =	
3 * 9 =	
3 * 10 =	

* 4	
4 * 1 =	
4 * 2 =	
4 * 3 =	
4 * 4 =	
4 * 5 =	
4 * 6 =	
4 * 7 =	
4 * 8 =	
4 * 9 =	
4 * 10 =	

* 5	
5 * 1 =	
5 * 2 =	
5 * 3 =	
5 * 4 =	
5 * 5 =	
5 * 6 =	
5 * 7 =	
5 * 8 =	
5 * 9 =	
5 * 10 =	

* 6	
6 * 1 =	
6 * 2 =	
6 * 3 =	
6 * 4 =	
6 * 5 =	
6 * 6 =	
6 * 7 =	
6 * 8 =	
6 * 9 =	
6 * 10 =	

* 7	
7 * 1 =	
7 * 2 =	
7 * 3 =	
7 * 4 =	
7 * 5 =	
7 * 6 =	
7 * 7 =	
7 * 8 =	
7 * 9 =	
7 * 10 =	

* 8	
8 * 1 =	
8 * 2 =	
8 * 3 =	
8 * 4 =	
8 * 5 =	
8 * 6 =	
8 * 7 =	
8 * 8 =	
8 * 9 =	
8 * 10 =	

* 9	
9 * 1 =	
9 * 2 =	
9 * 3 =	
9 * 4 =	
9 * 5 =	
9 * 6 =	
9 * 7 =	
9 * 8 =	
9 * 9 =	
9 * 10 =	

* 10	
10 * 1 =	
10 * 2 =	
10 * 3 =	
10 * 4 =	
10 * 5 =	
10 * 6 =	
10 * 7 =	
10 * 8 =	
10 * 9 =	
10 * 10 =	

* 1	
1 * 1 =	
1 * 2 =	
1 * 3 =	
1 * 4 =	
1 * 5 =	
1 * 6 =	
1 * 7 =	
1 * 8 =	
1 * 9 =	
1 * 10 =	

* 2	
2 * 1 =	
2 * 2 =	
2 * 3 =	
2 * 4 =	
2 * 5 =	
2 * 6 =	
2 * 7 =	
2 * 8 =	
2 * 9 =	
2 * 10 =	

* 3	
3 * 1 =	
3 * 2 =	
3 * 3 =	
3 * 4 =	
3 * 5 =	
3 * 6 =	
3 * 7 =	
3 * 8 =	
3 * 9 =	
3 * 10 =	

* 4	
4 * 1 =	
4 * 2 =	
4 * 3 =	
4 * 4 =	
4 * 5 =	
4 * 6 =	
4 * 7 =	
4 * 8 =	
4 * 9 =	
4 * 10 =	

* 5	
5 * 1 =	
5 * 2 =	
5 * 3 =	
5 * 4 =	
5 * 5 =	
5 * 6 =	
5 * 7 =	
5 * 8 =	
5 * 9 =	
5 * 10 =	

* 6	
6 * 1 =	
6 * 2 =	
6 * 3 =	
6 * 4 =	
6 * 5 =	
6 * 6 =	
6 * 7 =	
6 * 8 =	
6 * 9 =	
6 * 10 =	

* 7	
7 * 1 =	
7 * 2 =	
7 * 3 =	
7 * 4 =	
7 * 5 =	
7 * 6 =	
7 * 7 =	
7 * 8 =	
7 * 9 =	
7 * 10 =	

* 8	
8 * 1 =	
8 * 2 =	
8 * 3 =	
8 * 4 =	
8 * 5 =	
8 * 6 =	
8 * 7 =	
8 * 8 =	
8 * 9 =	
8 * 10 =	

* 9	
9 * 1 =	
9 * 2 =	
9 * 3 =	
9 * 4 =	
9 * 5 =	
9 * 6 =	
9 * 7 =	
9 * 8 =	
9 * 9 =	
9 * 10 =	

* 10	
10 * 1 =	
10 * 2 =	
10 * 3 =	
10 * 4 =	
10 * 5 =	
10 * 6 =	
10 * 7 =	
10 * 8 =	
10 * 9 =	
10 * 10 =	

* 1	
1 * 1 =	
1 * 2 =	
1 * 3 =	
1 * 4 =	
1 * 5 =	
1 * 6 =	
1 * 7 =	
1 * 8 =	
1 * 9 =	
1 * 10 =	

* 2	
2 * 1 =	
2 * 2 =	
2 * 3 =	
2 * 4 =	
2 * 5 =	
2 * 6 =	
2 * 7 =	
2 * 8 =	
2 * 9 =	
2 * 10 =	

* 3	
3 * 1 =	
3 * 2 =	
3 * 3 =	
3 * 4 =	
3 * 5 =	
3 * 6 =	
3 * 7 =	
3 * 8 =	
3 * 9 =	
3 * 10 =	

* 4	
4 * 1 =	
4 * 2 =	
4 * 3 =	
4 * 4 =	
4 * 5 =	
4 * 6 =	
4 * 7 =	
4 * 8 =	
4 * 9 =	
4 * 10 =	

* 5	
5 * 1 =	
5 * 2 =	
5 * 3 =	
5 * 4 =	
5 * 5 =	
5 * 6 =	
5 * 7 =	
5 * 8 =	
5 * 9 =	
5 * 10 =	

* 6	
6 * 1 =	
6 * 2 =	
6 * 3 =	
6 * 4 =	
6 * 5 =	
6 * 6 =	
6 * 7 =	
6 * 8 =	
6 * 9 =	
6 * 10 =	

* 7	
7 * 1 =	
7 * 2 =	
7 * 3 =	
7 * 4 =	
7 * 5 =	
7 * 6 =	
7 * 7 =	
7 * 8 =	
7 * 9 =	
7 * 10 =	

* 8	
8 * 1 =	
8 * 2 =	
8 * 3 =	
8 * 4 =	
8 * 5 =	
8 * 6 =	
8 * 7 =	
8 * 8 =	
8 * 9 =	
8 * 10 =	

* 9	
9 * 1 =	
9 * 2 =	
9 * 3 =	
9 * 4 =	
9 * 5 =	
9 * 6 =	
9 * 7 =	
9 * 8 =	
9 * 9 =	
9 * 10 =	

* 10	
10 * 1 =	
10 * 2 =	
10 * 3 =	
10 * 4 =	
10 * 5 =	
10 * 6 =	
10 * 7 =	
10 * 8 =	
10 * 9 =	
10 * 10 =	

* 1	
1 * 1 =	
1 * 2 =	
1 * 3 =	
1 * 4 =	
1 * 5 =	
1 * 6 =	
1 * 7 =	
1 * 8 =	
1 * 9 =	
1 * 10 =	

* 2	
2 * 1 =	
2 * 2 =	
2 * 3 =	
2 * 4 =	
2 * 5 =	
2 * 6 =	
2 * 7 =	
2 * 8 =	
2 * 9 =	
2 * 10 =	

* 3	
3 * 1 =	
3 * 2 =	
3 * 3 =	
3 * 4 =	
3 * 5 =	
3 * 6 =	
3 * 7 =	
3 * 8 =	
3 * 9 =	
3 * 10 =	

* 4	
4 * 1 =	
4 * 2 =	
4 * 3 =	
4 * 4 =	
4 * 5 =	
4 * 6 =	
4 * 7 =	
4 * 8 =	
4 * 9 =	
4 * 10 =	

* 5	
5 * 1 =	
5 * 2 =	
5 * 3 =	
5 * 4 =	
5 * 5 =	
5 * 6 =	
5 * 7 =	
5 * 8 =	
5 * 9 =	
5 * 10 =	

* 6	
6 * 1 =	
6 * 2 =	
6 * 3 =	
6 * 4 =	
6 * 5 =	
6 * 6 =	
6 * 7 =	
6 * 8 =	
6 * 9 =	
6 * 10 =	

* 7	
7 * 1 =	
7 * 2 =	
7 * 3 =	
7 * 4 =	
7 * 5 =	
7 * 6 =	
7 * 7 =	
7 * 8 =	
7 * 9 =	
7 * 10 =	

* 8	
8 * 1 =	
8 * 2 =	
8 * 3 =	
8 * 4 =	
8 * 5 =	
8 * 6 =	
8 * 7 =	
8 * 8 =	
8 * 9 =	
8 * 10 =	

* 9	
9 * 1 =	
9 * 2 =	
9 * 3 =	
9 * 4 =	
9 * 5 =	
9 * 6 =	
9 * 7 =	
9 * 8 =	
9 * 9 =	
9 * 10 =	

* 10	
10 * 1 =	
10 * 2 =	
10 * 3 =	
10 * 4 =	
10 * 5 =	
10 * 6 =	
10 * 7 =	
10 * 8 =	
10 * 9 =	
10 * 10 =	

* 1	
1 * 1 =	
1 * 2 =	
1 * 3 =	
1 * 4 =	
1 * 5 =	
1 * 6 =	
1 * 7 =	
1 * 8 =	
1 * 9 =	
1 * 10 =	

* 2	
2 * 1 =	
2 * 2 =	
2 * 3 =	
2 * 4 =	
2 * 5 =	
2 * 6 =	
2 * 7 =	
2 * 8 =	
2 * 9 =	
2 * 10 =	

* 3	
3 * 1 =	
3 * 2 =	
3 * 3 =	
3 * 4 =	
3 * 5 =	
3 * 6 =	
3 * 7 =	
3 * 8 =	
3 * 9 =	
3 * 10 =	

* 4	
4 * 1 =	
4 * 2 =	
4 * 3 =	
4 * 4 =	
4 * 5 =	
4 * 6 =	
4 * 7 =	
4 * 8 =	
4 * 9 =	
4 * 10 =	

* 5	
5 * 1 =	
5 * 2 =	
5 * 3 =	
5 * 4 =	
5 * 5 =	
5 * 6 =	
5 * 7 =	
5 * 8 =	
5 * 9 =	
5 * 10 =	

* 6	
6 * 1 =	
6 * 2 =	
6 * 3 =	
6 * 4 =	
6 * 5 =	
6 * 6 =	
6 * 7 =	
6 * 8 =	
6 * 9 =	
6 * 10 =	

* 7	
7 * 1 =	
7 * 2 =	
7 * 3 =	
7 * 4 =	
7 * 5 =	
7 * 6 =	
7 * 7 =	
7 * 8 =	
7 * 9 =	
7 * 10 =	

* 8	
8 * 1 =	
8 * 2 =	
8 * 3 =	
8 * 4 =	
8 * 5 =	
8 * 6 =	
8 * 7 =	
8 * 8 =	
8 * 9 =	
8 * 10 =	

* 9	
9 * 1 =	
9 * 2 =	
9 * 3 =	
9 * 4 =	
9 * 5 =	
9 * 6 =	
9 * 7 =	
9 * 8 =	
9 * 9 =	
9 * 10 =	

* 10	
10 * 1 =	
10 * 2 =	
10 * 3 =	
10 * 4 =	
10 * 5 =	
10 * 6 =	
10 * 7 =	
10 * 8 =	
10 * 9 =	
10 * 10 =	

Letras griegas

1 Alfabeto

Nombre																								
Mayúscula																								
Minúscula																								
Nombre																								
Mayúscula																								
Minúscula																								
Total letras																								

2 Abecedario

Mayúscula																										
Minúscula																										
Total letras																										

3 Alfabeto

Nombre	Alfa	Beta	Gamma	Delta	Epsilon	Zeta	eta	theta	iota	Kappa	Lambda	Mu	Nu	Xi	Omicron	Pi	Rho	sigma	Tau	Ipsilon	Fi	Ji	Psi	Omega
Mayúscula	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Minúscula	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω
Total letras	24																							

4 Abecedario

Mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Total letras	27																										

1 Alfabeto

Nombre																								
Mayúscula																								
Minúscula																								
Nombre																								
Mayúscula																								
Minúscula																								
Total letras																								

2 Abecedario

Mayúscula																										
Minúscula																										
Total letras																										

3 Alfabeto

Nombre	Alfa	Beta	Gamma	Delta	Epsilon	Zeta	eta	theta	iota	Kappa	Lambda	Mu	Nu	Xi	Omicron	Pi	Rho	sigma	Tau	Ipsilon	Fi	Ji	Psi	Omega
Mayúscula	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Minúscula	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω
Total letras	24																							

4 Abecedario

Mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Total letras	27																										

1 Alfabeto

Nombre																								
Mayúscula																								
Minúscula																								
Nombre																								
Mayúscula																								
Minúscula																								
Total letras																								

2 Abecedario

Mayúscula																										
Minúscula																										
Total letras																										

3 Alfabeto

Nombre	Alfa	Beta	Gamma	Delta	Epsilon	Zeta	eta	theta	iota	Kappa	Lambda	Mu	Nu	Xi	Omicron	Pi	Rho	sigma	Tau	Ipsilon	Fi	Ji	Psi	Omega
Mayúscula	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Minúscula	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω
Total letras	24																							

4 Abecedario

Mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Total letras	27																										

1 Alfabeto

Nombre																								
Mayúscula																								
Minúscula																								
Nombre																								
Mayúscula																								
Minúscula																								
Total letras																								

2 Abecedario

Mayúscula																										
Minúscula																										
Total letras																										

3 Alfabeto

Nombre	Alfa	Beta	Gamma	Delta	Epsilon	Zeta	eta	theta	iota	Kappa	Lambda	Mu	Nu	Xi	Omicron	Pi	Rho	sigma	Tau	Ipsilon	Fi	Ji	Psi	Omega
Mayúscula	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Minúscula	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω
Total letras	24																							

4 Abecedario

Mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Total letras	27																										

1 Alfabeto

Nombre																								
Mayúscula																								
Minúscula																								
Nombre																								
Mayúscula																								
Minúscula																								
Total letras																								

2 Abecedario

Mayúscula																										
Minúscula																										
Total letras																										

3 Alfabeto

Nombre	Alfa	Beta	Gamma	Delta	Epsilon	Zeta	eta	theta	iota	Kappa	Lambda	Mu	Nu	Xi	Omicron	Pi	Rho	sigma	Tau	Ipsilon	Fi	Ji	Psi	Omega
Mayúscula	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Minúscula	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω
Total letras	24																							

4 Abecedario

Mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Total letras	27																										

1 Alfabeto

Nombre																								
Mayúscula																								
Minúscula																								
Nombre																								
Mayúscula																								
Minúscula																								
Total letras																								

2 Abecedario

Mayúscula																										
Minúscula																										
Total letras																										

3 Alfabeto

Nombre	Alfa	Beta	Gamma	Delta	Epsilon	Zeta	eta	theta	iota	Kappa	Lambda	Mu	Nu	Xi	Omicron	Pi	Rho	sigma	Tau	Ipsilon	Fi	Ji	Psi	Omega
Mayúscula	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Minúscula	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω
Total letras	24																							

4 Abecedario

Mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Total letras	27																										

1 Alfabeto

Nombre																								
Mayúscula																								
Minúscula																								
Nombre																								
Mayúscula																								
Minúscula																								
Total letras																								

2 Abecedario

Mayúscula																										
Minúscula																										
Total letras																										

3 Alfabeto

Nombre	Alfa	Beta	Gamma	Delta	Epsilon	Zeta	eta	theta	iota	Kappa	Lambda	Mu	Nu	Xi	Omicron	Pi	Rho	sigma	Tau	Ipsilon	Fi	Ji	Psi	Omega
Mayúscula	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Minúscula	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω
Total letras	24																							

4 Abecedario

Mayúscula	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Minúscula	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Total letras	27																										

Derivadas e Integrales

1

Derivadas

Aprendamos reglas de derivación.
Consideremos ***a***,***b*** y ***k*** como constantes y números reales, tambien consideremos ***u*** y ***v*** como funciones.

Derivada de una constante

$f(x) = k$	$f'(x) = 0$

Derivada de una variable

$f(x) = x$	$f'(x) = 1$

Derivada de una función lineal

$f(x) = ax + b$	$f'(x) = a$

Derivada de una potencia

$f(x) = u^k$	$f'(x) = k * u^{k-1} * u'$

Ejemplos

$f(x) = -2$	$f'(x) = 0$
$f(x) = -5x$	$f'(x) = -5$
$f(x) = -\frac{7}{2}x - 3$	$f'(x) = -\frac{7}{2}$
$f(x) = x^4$	$f'(x) = 4x^3$

Derivadas

$f(x) = x^{-4}$	$f'(x) = -4x^{-5} = -\frac{4}{x^5}$
$f(x) = (2x - \frac{1}{2})^3$	$f'(x) = 3(2x - \frac{1}{2})^2 * 2$

1

Derivadas

Aprendamos reglas de derivación.
Consideremos ***a***,***b*** y ***k*** como constantes y números reales, tambien consideremos ***u*** y ***v*** como funciones.

Derivada de una constante

$f(x) = k$	$f'(x) =$

Derivada de una variable

$f(x) = x$	$f'(x) =$

Derivada de una función lineal

$f(x) = ax + b$	$f'(x) =$

Derivada de una potencia

$f(x) = u^k$	$f'(x) =$

Ejemplos

$f(x) = -2$	$f'(x) =$
$f(x) = -5x$	$f'(x) =$
$f(x) = -\frac{7}{2}x - 3$	$f'(x) =$
$f(x) = x^4$	$f'(x) =$

Derivadas

$f(x) = x^{-4}$	$f'(x) =$
$f(x) = (2x - \frac{1}{2})^3$	$f'(x) =$

1

Derivadas

Aprendamos reglas de derivación.
Consideremos ***a***,***b*** y ***k*** como constantes y números reales, tambien consideremos ***u*** y ***v*** como funciones.

Derivada de una constante

$f(x) = k$	$f'(x) =$

Derivada de una variable

$f(x) = x$	$f'(x) =$

Derivada de una función lineal

$f(x) = ax + b$	$f'(x) =$

Derivada de una potencia

$f(x) = u^k$	$f'(x) =$

Ejemplos

$f(x) = -2$	$f'(x) =$
$f(x) = -5x$	$f'(x) =$
$f(x) = -\frac{7}{2}x - 3$	$f'(x) =$
$f(x) = x^4$	$f'(x) =$

Derivadas

$f(x) = x^{-4}$	$f'(x) =$
$f(x) = (2x - \frac{1}{2})^3$	$f'(x) =$

1

Derivadas

Aprendamos reglas de derivación.
Consideremos ***a***,***b*** y ***k*** como constantes y números reales, tambien consideremos ***u*** y ***v*** como funciones.

Derivada de una constante

$f(x) = k$	$f'(x) =$

Derivada de una variable

$f(x) = x$	$f'(x) =$

Derivada de una función lineal

$f(x) = ax + b$	$f'(x) =$

Derivada de una potencia

$f(x) = u^k$	$f'(x) =$

Ejemplos

$f(x) = -2$	$f'(x) =$
$f(x) = -5x$	$f'(x) =$
$f(x) = -\frac{7}{2}x - 3$	$f'(x) =$
$f(x) = x^4$	$f'(x) =$

Derivadas

$f(x) = x^{-4}$	$f'(x) =$
$f(x) = (2x - \frac{1}{2})^3$	$f'(x) =$

1

Derivadas

Aprendamos reglas de derivación.
Consideremos ***a***,***b*** y ***k*** como constantes y números reales, tambien consideremos ***u*** y ***v*** como funciones.

Derivada de una constante

$f(x) = k$	$f'(x) =$

Derivada de una variable

$f(x) = x$	$f'(x) =$

Derivada de una función lineal

$f(x) = ax + b$	$f'(x) =$

Derivada de una potencia

$f(x) = u^k$	$f'(x) =$

Ejemplos

$f(x) = -2$	$f'(x) =$
$f(x) = -5x$	$f'(x) =$
$f(x) = -\frac{7}{2}x - 3$	$f'(x) =$
$f(x) = x^4$	$f'(x) =$

Derivadas

$f(x) = x^{-4}$	$f'(x) =$
$f(x) = (2x - \frac{1}{2})^3$	$f'(x) =$

1

Integrales

Aprendamos integrales inmediatas.
Consideremos ***a***,***k*** y ***c*** como constantes y números reales, tambien consideremos ***u*** como funcion y ***'u*** derivada de **u**.

Integral de una constante

$\int k dx$	$k * x + C$

Integral de cero

$\int 0 dx$	C

Integral de una potencia

$\int x^n dx$	$\frac{x^{n+1}}{n+1} + C$
$\int u^n * u' dx$	$\frac{u^{n+1}}{n+1} + C$

Integral de una constante por función

$\int kx^n dx$	$k \int x^n dx$	$k * \frac{x^{n+1}}{n+1}$

Derivadas

Ejemplos

$\int 7 dx$	$7x + C$		
$\int x^6 dx$	$\frac{x^7}{7} + C$		
$\int 7x^3 dx$	$\frac{7x^4}{4} + C$		
$\int \frac{3}{x^4} dx$	$\int 3x^{-4} dx$	$\frac{3x^{-3}}{-3} + C$	$\frac{1}{x^3} + C$
$\int 2x dx$	$2 \int x^1 dx$	$2 * \frac{x^2}{2}$	x^2

1

Integrales

Aprendamos integrales inmediatas.
Consideremos ***a***,***k*** y ***c*** como constantes y números reales, tambien consideremos ***u*** como funcion y ***'u*** derivada de **u**.

Integral de una constante

$\int k dx$	

Integral de cero

$\int 0 dx$	

Integral de una potencia

$\int x^n dx$	
$\int u^n * u' dx$	

Integral de una constante por función

$\int k x^n dx$	

Derivadas

Ejemplos

$\int 7 dx$			
$\int x^6 dx$			
$\int 7x^3 dx$			
$\int \frac{3}{x^4} dx$			
$\int 2x dx$			

1

Integrales

Aprendamos integrales inmediatas.
Consideremos ***a***,***k*** y ***c*** como constantes y números reales, tambien consideremos ***u*** como funcion y ***'u*** derivada de **u**.

Integral de una constante

$\int k dx$	

Integral de cero

$\int 0 dx$	

Integral de una potencia

$\int x^n dx$	
$\int u^n * u' dx$	

Integral de una constante por función

$\int k x^n dx$	

Derivadas

Ejemplos

$\int 7 dx$			
$\int x^6 dx$			
$\int 7x^3 dx$			
$\int \frac{3}{x^4} dx$			
$\int 2x dx$			

1

Integrales

Aprendamos integrales inmediatas.
Consideremos ***a***,***k*** y ***c*** como constantes y números reales, tambien consideremos ***u*** como funcion y ***'u*** derivada de **u**.

Integral de una constante

$\int k dx$	

Integral de cero

$\int 0 dx$	

Integral de una potencia

$\int x^n dx$	
$\int u^n * u' dx$	

Integral de una constante por función

$\int k x^n dx$	

Derivadas

Ejemplos

$\int 7 dx$			
$\int x^6 dx$			
$\int 7x^3 dx$			
$\int \frac{3}{x^4} dx$			
$\int 2x dx$			

1

Integrales

Aprendamos integrales inmediatas.
Consideremos ***a***,***k*** y ***c*** como constantes y números reales, tambien consideremos ***u*** como funcion y ***'u*** derivada de **u**.

Integral de una constante

$\int k dx$	

Integral de cero

$\int 0 dx$	

Integral de una potencia

$\int x^n dx$	
$\int u^n * u' dx$	

Integral de una constante por función

$\int k x^n dx$	

Derivadas

Ejemplos

$\int 7 dx$			
$\int x^6 dx$			
$\int 7x^3 dx$			
$\int \frac{3}{x^4} dx$			
$\int 2x dx$			