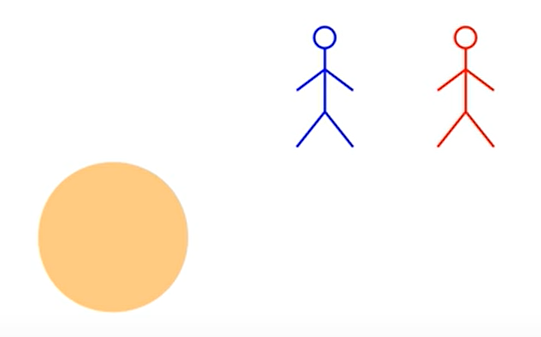
Assignment Cut and split

אלגוריתם לחילוק עוגה בין שתי אנשים

In this assignment we will develop a Software for, **fairly cutting process** for 2 -> n persons.

**How to split a cake fairly between 2 persons?**



2 persons! how can we make a cut that will be fairly for each of them?

**סיפור חלוקה כך שהתוצאות לא יהיו שוות**

Unfair Situation

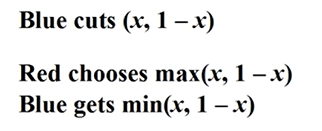
* Person 1 cuts and choose.  
  for his purpose (person 1) won’t make the cut equal so he could get the largest size of the cake for his beneficial.

**סיפור חלוקה שווה בין שתי הגורמים**

Fair algorithm 2 Persons:

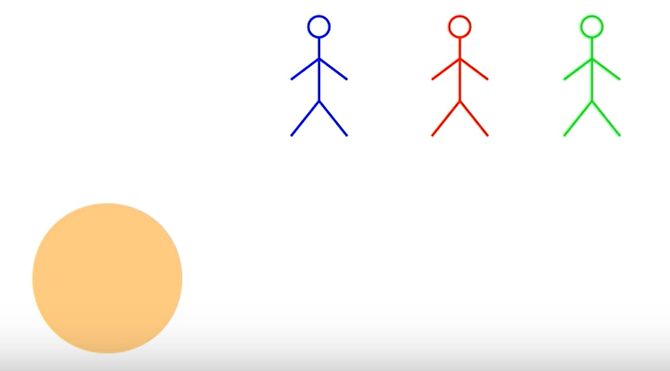
* Person 1 cuts the cake the way he wishes.
* Person 2 will inspect the slices and choose witch one he wants.
* The last slice will go to person 1.

**This algorithm will solve fairly slice between 2 persons.**  
Explanation: person 1 cuts the cake, will make the cut as even he could.  
**why?** if he doesn’t in all the cases he will loose and get the smallest portion of the cake, the person 2 will choose the largest slice and person 1 will have the small portion.

Algorithm conclusion:

אלגוריתם לחילוק עוגה בין שלושה אנשים

**How to split a cake fairly between 3 persons?**



3 persons! how can we make a cut that will be fairly for each of them?

**נתאר סיפור חלוקה שווה בין שלושת הגורמים**

Fair algorithm for 3 Persons:

* Person 1 cuts the cake the way he wishes.
* Person 2 will inspect the slices and choose if he wants to cut the pieces more.
* The last persons that cuts the slice will get it.

This algorithm will solve the fairly slice between 3 persons.  
Explanation: person 1 and 2 cuts the cake, they will make the cut as even they could 1/3.  
**why?** if they don’t in all the cases the 3 person will not cut and get the largest portion of the cake.

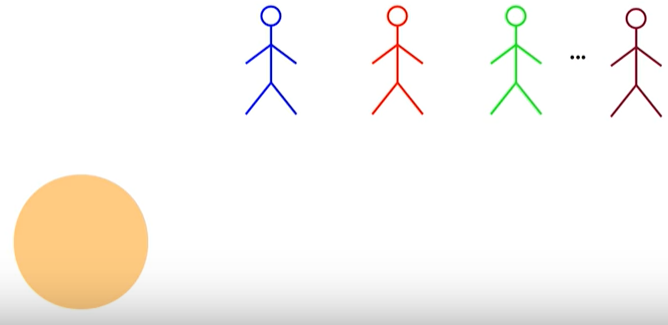
***Note that if someone cuts the cake to small than a 1/3, he will get that slice.***

When a person has received the slice. We are left with two pieces of the cake and 2 persons.

To solve this situation, we do it with the algorithm for 2 persons.

אלגוריתם לחילוק עוגה בין אינסוף אנשים

**Now that we have a solution for 2 and 3 persons, we can create an algorithm for N persons.**



**נתאר סיפור חלוקה שווה בין אינסוף גורמים**

**Algorithm for N persons.**

* Person 1 cuts the slice
* Person 2, 3…N have a choice to cut off some of the slices, the last person that cut a piece gets that slice.
* First slice gets 1/N
* Repeat this algorithm for N-1 persons.
* For final 2 persons remaining will do the cut and choose algorithm.

**Coding time**

Now that we understand the algorithm’s how they work. we must write the software for it.

Agents

An abstract class that describes a participant in a cake-cutting algorithm.

\_\_init\_\_(self, name:str=None)

Constructor for initialing name of agent

name(self)

function that returns name of agent

Abstract methodes:

cake\_value(self):

returns value of the cake for agent

cake\_length(self):

returns the length of the total cake that the agent wants

eval(self, start:float, end:float):

returns the value of the starting and ending point of the slice agent wants

mark(self, start:float, targetValue:float):

return "end" such that the value of the interval [start,end] is targetValue.

piece\_value(self, piece:List[tuple]):

returns the sum value of multiple slices.

partition\_values(self, partition:List[float]):

function that evaluates all the partitions.

cut\_and\_choose

Implementation of the last diminisher protocol for fair cake-cutting among 2 agents.

asymmetric\_protocol(agents: List[Agent])->Allocation:

one cuts and the other chooses.

a list that must contain exactly 2 Agent objects.

symmetric\_protocol(agents: List[Agent])->Allocation:

both agents cut, the manager chooses who gets what.

a list that must contain exactly 2 Agent objects.

last\_diminisher

Implementation of the last diminisher protocol for fair cake-cutting among n agents.

last\_diminisher(agents: List[Agent])->Allocation:

list of Agent objects.

return: a proportional cake-allocation.

last\_diminisher\_recursive(start:float, agents: List[Agent], active\_agents:List[int], allocation:Allocation):

A recursive subroutine for last-diminisher.

:param start: the leftmost end of the cake that should be allocated.

:param agents: the list of all n agents in the original protocol.

:param active\_agents: list of indices of those agents who are still active (not allocated yet).

:param allocation: the current allocation (will be updated during the run).

:return: nothing - the allocation is modified in place.

**By:  
Adam Lev Ari 204115547**

**Nathan Almaleh 209633635**