* Intro : Qu’est-ce le jeu ‘Katamino’?
  + Les polyominos
    - Dénomination/représentation
    - Historique
* Objectif du TIPE – précision sur les Katamino
  + Rappel de la pbt
* Notre solveur pour polyomino
  + Première idée

Katamino est un jeu dont le but est de remplir une grille de largeur 5 et de longueur 3 à 13 avec des pièces de formes diverses appelées kataminos.

Notre TIPE a pour but de créer un résolveur qui à une liste de kataminos et une grille vide associe une grille résolue et d’en déduire des pistes d’optimisation.

Les Kataminos s’inscrivent plus largement dans la catégorie des polyominos.

On peut les définir de plusieurs façons, nous allons utiliser celle dite des « polyominos à forme libres » ou PFL. Dans cette représentation un polyomino est composée de lui-même ainsi que de ses symétries centrales et axiales.

« La figure 2 représente un seul polyomino » (figure 2)

Le grand public connait les tetrominos de Tetris, les pentaminos de Blokus ou les dominos

Le terme a été inventé par le mathématicien S W Golomb en 1953 qui y dédia plusieurs livres (notamment : Polyominoes : Puzzles, Pattern, Problem, and Packing ) et de nombreux problèmes ludiques publiés dans la rubrique de Martin Gardner « Mathematical Games » dans le journal Scientific America.

Aujourd’hui encore, on peut trouver des problèmes de polyominos dans (toute bonne revue scientifique qui se respecte) de nombreux livres et championnats de mathématiques récréatives.

La représentation la plus courante est celle sous forme de carrés unitaires joints par les côtés.

On peut également représenter une pièce sous forme d’une matrice et donc une grille résolue également.

Il existe différentes méthodes pour remplir cette grille. Commençons par le résolveur itératif.

-**Résolveur itératif** consiste à tester toutes les possibilités et à s’arrêter lorsque qu’il en a trouvé une.

« Explication du logigramme ….. » figure 8

-Etude de la complexité

Truc d’Andréas

Modèle de comptage de la complexité

Poser un incrément qui augmente à chaque coup constant

Validité du modèle ?

Pas vraiment possible de tirer une conclusion

-Algorithme glouton :

Il semblerait qu’il vaille mieux placer les plus grosses pièces en premières. Vérifions cette hypothèse.

En globalité ça parait bien en zoomant on remarque que c’est moins vrai pour des grilles avec une complexité < 1e6 (moins de 2 secondes)

On peut dire que cette hypothèse est validée

import time

Ck=[]

Ct=[]

Liste20=open('combi - Copie/test20.txt','r')

Listes20=Liste20.readlines()

Liste20.close()

for i in range (len(Listes20)):

Listes20[i]=eval(Listes20[i].strip())

for i in range(len(Listes20)):

tri\_insertion(Listes20[i])

compl=0

force\_brute(Listes20[i])

Ck.append(compl)

print(i,' /',len(Listes20),':\n',compl)

Ct=[[] for \_ in range(len(Listes20))]

for k in range (100):

for i in range (len(Listes20)):

print('k : ',k,'/ 100 \n i : ',i,' /',len(Listes20),' \n \n')

t1=time.time()

force\_brute(Listes20[i])

t2=time.time()

Ct[i].append(t2-t1)

print("0",t1,t2,t2-t1)

t1=time.time()

force\_brute(Listes20[i],1)

t2=time.time()

Ct[i].append(t2-t1)

print("1",t1,t2,t2-t1)

plt.plot(Ck,Ct,'\*')

a,b=np.polyfit(Ck,Ct,1)

x=np.linspace(0, max(Ct),2000)

plt.plot(x,a\*x+b)

plt.show()

Ct2=[]

for i in range (len(Ct)):

Ct2.append(np.mean(Ct[i]))