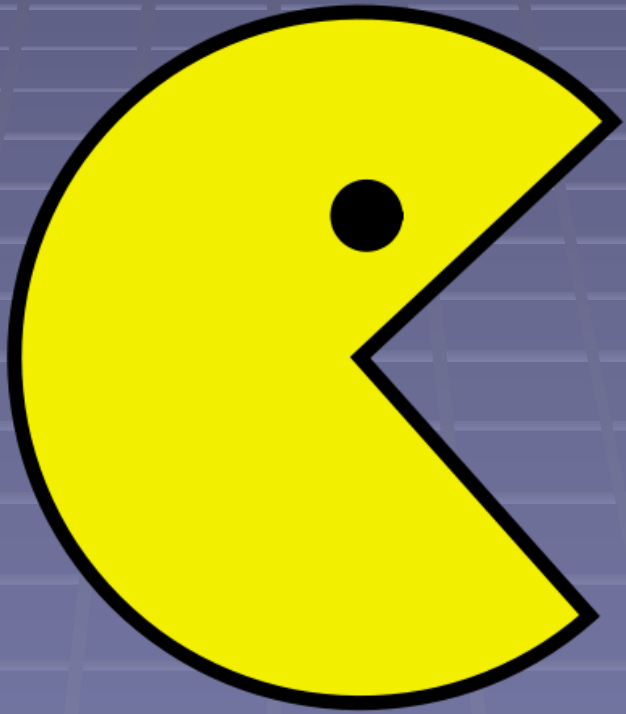
# Heuristiques gloutonnes



Jean-Philippe Préaux

http://www.i2m.univ-amu.fr/~preaux

# Problème du sac à dos

- n objets disponibles.
- L'objet i a un poids $p_i>0$ et une valeur $v_i>0$.
- Emporter un sous-ensemble d'objets de valeur maximale dans un sac de capacité P.
- Dans le <u>problème du sac à dos 0-1</u>, chaque objet est en unique exemplaire.
- $\Sigma_i \, p_i > P$ sinon le problème est trivial.

# Problème du sac à dos

- n objets disponibles.
- L'objet i a un poids $p_i>0$ et une valeur $v_i>0$.
- Emporter un sous-ensemble d'objets de valeur maximale dans un sac de capacité P.
- Dans le problème du sac à dos en nombres entiers chaque objet est en une infinité d'exemplaire.

# Heuristique gloutonne

<u>Pour le sac à dos en 0-1 :</u>

- Numéroter les objets i par $v_i/p_i$ décroissants.
- Poids=0 ; valeur=0
- Pour i variant de 1 jusqu'à n
  - Si Poids + $p_i \leq$ P alors
    - Poids=Poids+$p_i$
    - Valeur=Valeur+$v_i$
  - Fin Si
- Fin pour

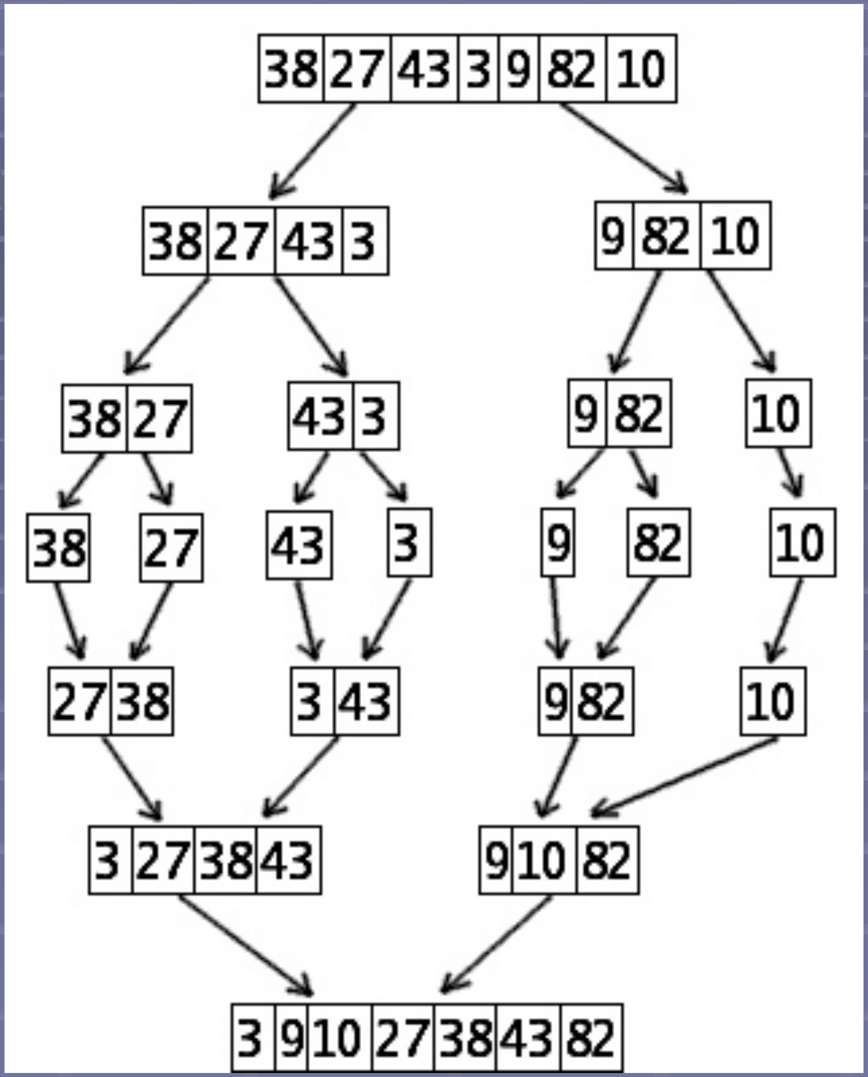
# Heuristique gloutonne

<u>Pour le sac à dos en nombres entiers :</u>

- Numéroter les objets i par $v_i/p_i$ décroissants.
- Poids=0 ; valeur=0
- Pour i variant de 1 jusqu'à n
  - Tant que Poids + $p_i \leq P$
    - Poids=Poids+$p_i$
    - Valeur=Valeur+$v_i$
  - Fin Tant que
- Fin pour

- Une heuristique gloutonne consistant à prendre d'abord les objets les plus chers, ou les plus lourds n'est pas très bonne : elle ignore une partie des données.
- La complexité de l'heuristique est en $O(n^2)$ avec un algorithme de tri naïf (tri à bulle, tri par insertion,...). En $O(n \log n)$ avec un algorithme de tri efficace (tri fusion, tri par tas...)

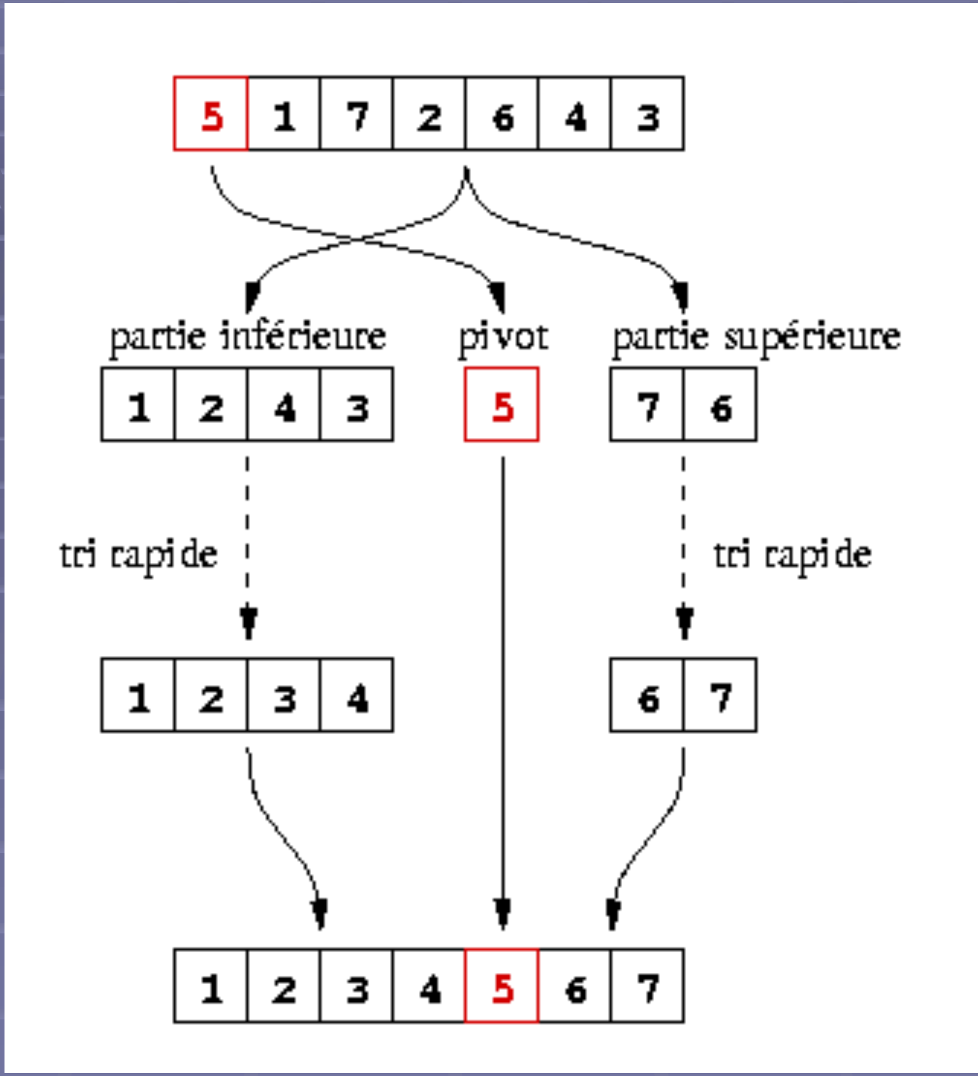
# TRI FUSION




Complexité : O(n log n)

Principe fondamental : DIVISER POUR REGNER !

# TRI RAPIDE




Complexité $O(n^2)$, en moyenne $O(n \log n)$.

Complexité $O(n \log n)$ si l'on choisit bien le pivot (introsort...)

- Pour le sac à dos en nombre entiers l'heuristique n'est pas mauvaise, de PRP 0,5.
- Exemple : $n=2$, $p_1=v_1=k+1$, $p_2=v_2=k$, $P=2k$. L'algorithme (peut) trouve(r) $k+1$ tandis que l'optimum est $2k$. Ainsi $P_H$ peut-être aussi proche que l'on souhaite de 0.5 ($k \to \infty$).
- Avec $M=P/\max(p_i)$, on montre que $P_H=M/(M+1)$ : plus les objets sont légers  meilleure est la performance de l'heuristique.

- L’heuristique est mauvaise dans le cas du problème du sac à dos en 0-1.
- Exemple : $n=2$, $p_1=v_1=1$, $p_2=v_2=P$.
- Elle (peut) trouve(r) 1 alors que l’optimum est P.
- Ainsi $P_H \to 0$ quand $P \to \infty$.
- Le comportement moyen est bon en moyenne.
- Sans utilité pour les problèmes d’empilement ($p_i=v_i$)

# Problème de mise en boîte

- Ranger n objets de tailles $a_i$, i=1,..,n, dans un nombre minimal de boîtes de capacité b.
- First-Fit (FF) : place à l'itération i l'objet i dans la première boîte qui convienne.
- Best-Fit (BF) : place à l'itération i l'objet i dans la boîte qu'il remplit le mieux.

Intuitivement il vaut mieux placer d'abord les objets les plus gros.

- Avec un tri préalable par ordre décroissant sur la taille des objets on obtient :
- First-Fit-Decreasing (FFD).
- Best-Fit-decreasing (BFD).

- FF et BF ont une PRP de 2.
- FFD et BFD ont une PRP de 1.5.
- BFD est meilleure que FFD en moyenne.
- On a prouvé des résultats + fins :
- FF(d) $\leq$ 17/10 OPT(d)+2 (tend vers 70%)
- FFD(d) $\leq$ 11/9 OPT(d)+4 (tend vers 22.2%)
- Ca marche mieux quand la taille diminue :
- Si $a_i \leq B/2$ la PRP tend vers 18.3%.
- Si $B/3<a_i<B$ c'est un problème de couplage, polynomial. Dans ce cas FFD donne toujours l'optimum.