Organisation

- le 13 décembre, cours d'algorithmique
- cours de prog le **jeudi 4 janvier** de 14h30 à 17h30

Gestion de la mémoire à l'exécution

- en C: `malloc, free`
- en Caml: *"toute valeur est éternelle"*
  - pas d'allocation explicite de mémoire
    (mais `ref` – et `new` – en font presque)
  - pas de désallocation explicite
- l'éternité ça prend de la place
  - certaines choses éternelles deviennent inutilisables
- gestion automatique de l'allocation dynamique

Glaneurs de cellules

Grandes familles de GC

Exploration du tas

- les algorithmes les plus courants ont pour principe d'examiner le tas pour y détecter les zones libérables
- qui sont les objets vivants à l'instant t?
  - variables globales, variables locales, appels courants des fonctions $\longrightarrow$ *ensemble des racines* (root set)
  - critère d'atteignabilité: les gens pointés par des gens vivants
  - approximation du caractère vivant: analyses à la compilation peuvent être plus précises
  - à ce sujet, il existe des méthodes d'analyse statique pour détecter qu'une donnée ne survit pas à la fonction qui l'a créée: allocation dans la pile

Marquer et balayer *(Mark & Sweep)*

- principe: deux phases
  - *reconnaître* à l'instant t l'ensemble des objets vivants
  - *libérer* les objets morts
- comment
  - à partir de l'ensemble des racines, marquer tous les objets atteignables (1 bit par objet en mémoire)
  - rassembler les objets non marqués dans une liste chaînée (cf. `malloc`) et démarquer les vivants
- problèmes:
  1 mémoire fragmentée
  2 coût proportionnel à la taille du tas
  3 localité mal gérée (cf. 1)
  4 une grande pause-GC
- algorithmes Mark-Compact
  - a priori encore plus lent: davantage de passes sur le tas

Comment reconnaître les pointeurs?

pointeur ou valeur immédiate?

- représentation des pointeurs en mémoire:
  - utilisation de quelques *bits d'étiquette*
  - en Caml: valeur sur le tas ou immédiate (`int`: 31 bits)
- selon les langages et les implémentations, plus ou moins d'information sur le type des objets en mémoire
  - contient ou non des pointeurs vers d'autres objets
  - si le langage est fortement typé, possibilité de ne pas étiqueter
    (l'ensemble des racines suffit, puis on "suit les types")
- approximation (algorithmes *conservatifs*) pas d'étiquette, on regarde si adresse valide, *alignée*

  ouvre la voie au GC pour langages sans GC (p.ex. C, en demandant au programmeur d'être gentil)