M1103 Amphi 5 : Complexité en temps

Thomas Nowak

Université Paris-Sud

La dernière fois

Complexité en temps

Mesures de qualité d'un programme

- mesure de qualité la plus importante = *correction* du programme
- parfois en conflit : *lisibilité* et *performance*
- performance en termes de temps d'exécution, de mémoire utilisée, de consommation d'énergie
- la performance est-elle importante avec des ordinateurs de plus en plus rapides et des espaces de mémoire de plus en plus grands ?
- parfois non, parfois oui : à l'échelle des masses de données de Google/Facebook certainement

Mesures de qualité d'un programme

- en absence d'une bonne raison pour améliorer la performance : maximiser la lisibilité
- car cela aide à garantir la correction du programme
  - bugs plus faciles à trouver
  - pas de cas de bords oubliés
  - etc.
- mais si on oublie complètement la performance, on peut payer très cher relatif à la performance optimale
- souvent il est possible de combiner la lisibilité *et* la performance
  - trouver "le bon" algorithme pour un problème

Mesurer le temps d'exécution

- comment mesurer le temps d'exécution d'un programme ?
  - avec une horloge
  - compter le nombre d'opérations
  - estimer l'ordre de grandeur du nombre d'opérations

Mesurer le temps d'exécution

- le temps réel d'exécution mesuré avec une horloge donne des résultats très exacts, mais :
  - varie d'un ordinateur à l'autre
  - varie d'une entrée à l'autre
  - varie même sur le même ordinateur avec les mêmes entrées à cause de facteurs externes (autres programmes qui tournent sur l'ordinateur par exemple)

Mesurer le temps d'exécution

- compter le nombre d'opérations donne des résultats indépendants de l'ordinateur et de facteurs externes, mais :
  - varie d'une entrée à l'autre
  - il n'est pas évident quelles opérations on doit compter

Exemple 1

```
double f2c(double f)
{
    return (f - 32.0) * 5.0/9.0;
}
```

Opérations à compter

- on suppose que les opérations suivantes ont des temps d'exécution comparables :
  - opérations mathématiques de base
  - comparaisons de types de base
  - affectation de types de base
  - accès à un élément d'un tableau ou d'un vecteur

Exemple 2

```
int f(int i)
{
    int ret = 1;

    while(i >= 1)
    {
        ret *= i;
        i--;
    }

    return ret;
}
```

Simulation d'algorithmes

- il peut être difficile de savoir combien de fois une boucle est exécutée
- pour une entrée fixe, on peut faire une *simulation* du programme (à la main)
- créer une table avec une colonne pour chaque variable et une ligne pour des instants de l'exécution
- la simulation avec quelques entrées peut donner une idée pour une formule générale pour le nombre d'itérations

Exemple 3

```
bool r(const vector<int> &vec, int x)
{
    for(int i = 0; i < vec.size(); ++i)
    {
        if(vec[i] == x)
            return true;
    }

    return false;
}
```

Dépendance des entrées

- le nombre d'opérations exécutées dépend fortement des entrées
- souvent, on l'exprime en la *taille* de l'entrée (ex : taille d'un entier `n`, nombre d'éléments d'un vecteur)
- et ceci pour l'entrée *pire cas* (contrairement au meilleur cas ou au cas moyen)

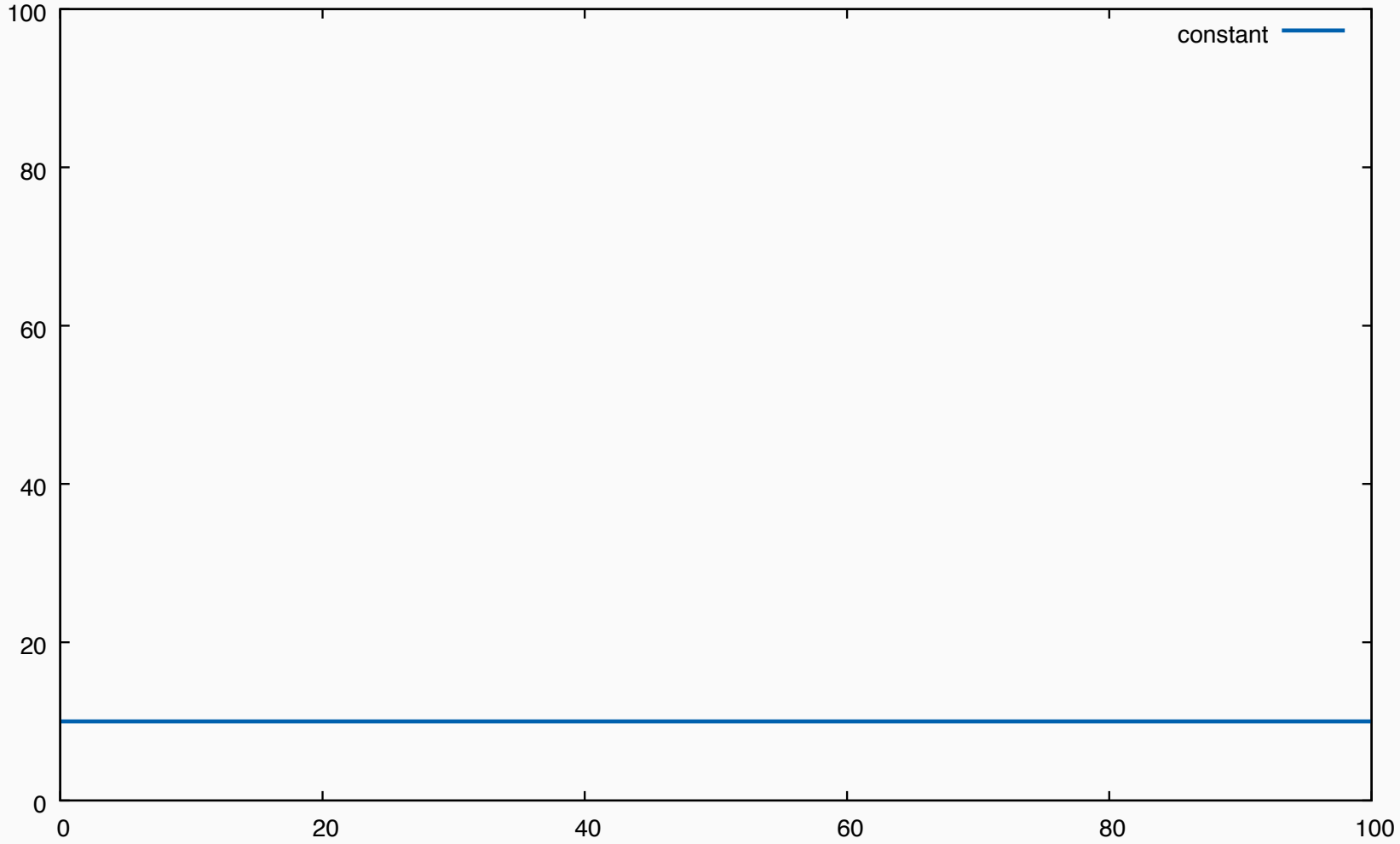
Notation de Landau

- le nombre exact d'opérations n'est pas toujours très intéressant
- ce qui est plus important est *l'ordre de grandeur*
- exemple : si on compte $43n + 13$ opérations, on peut dire que l'ordre de grandeur est *linéaire* (en n) parce que le terme dominant est le terme linéaire $43n$
- notation de Landau : $O(n)$ pour tout compte d'opérations dont l'ordre de grandeur est linéaire
- se généralise à d'autres ordres de grandeurs (ex : $O(1)$, $O(n^2)$, $O(2^n)$)

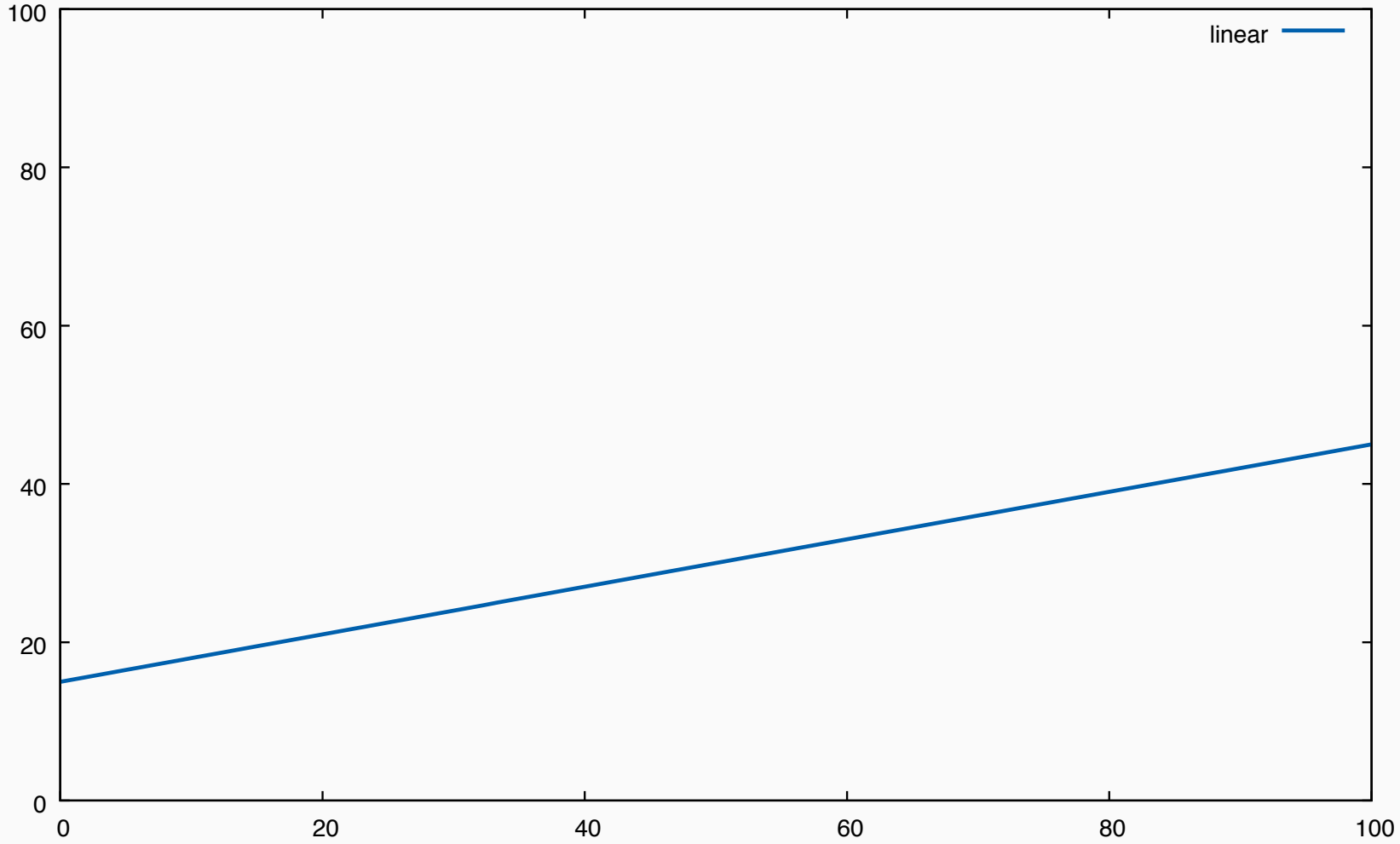
Classes de complexité

- on distingue souvent :
  - complexité constante $O(1)$
  - complexité linéaire $O(n)$
  - complexité polynomiale $O(n^k)$ pour un k donné
  - complexité exponentielle $O(2^n)$ (ou autre base)

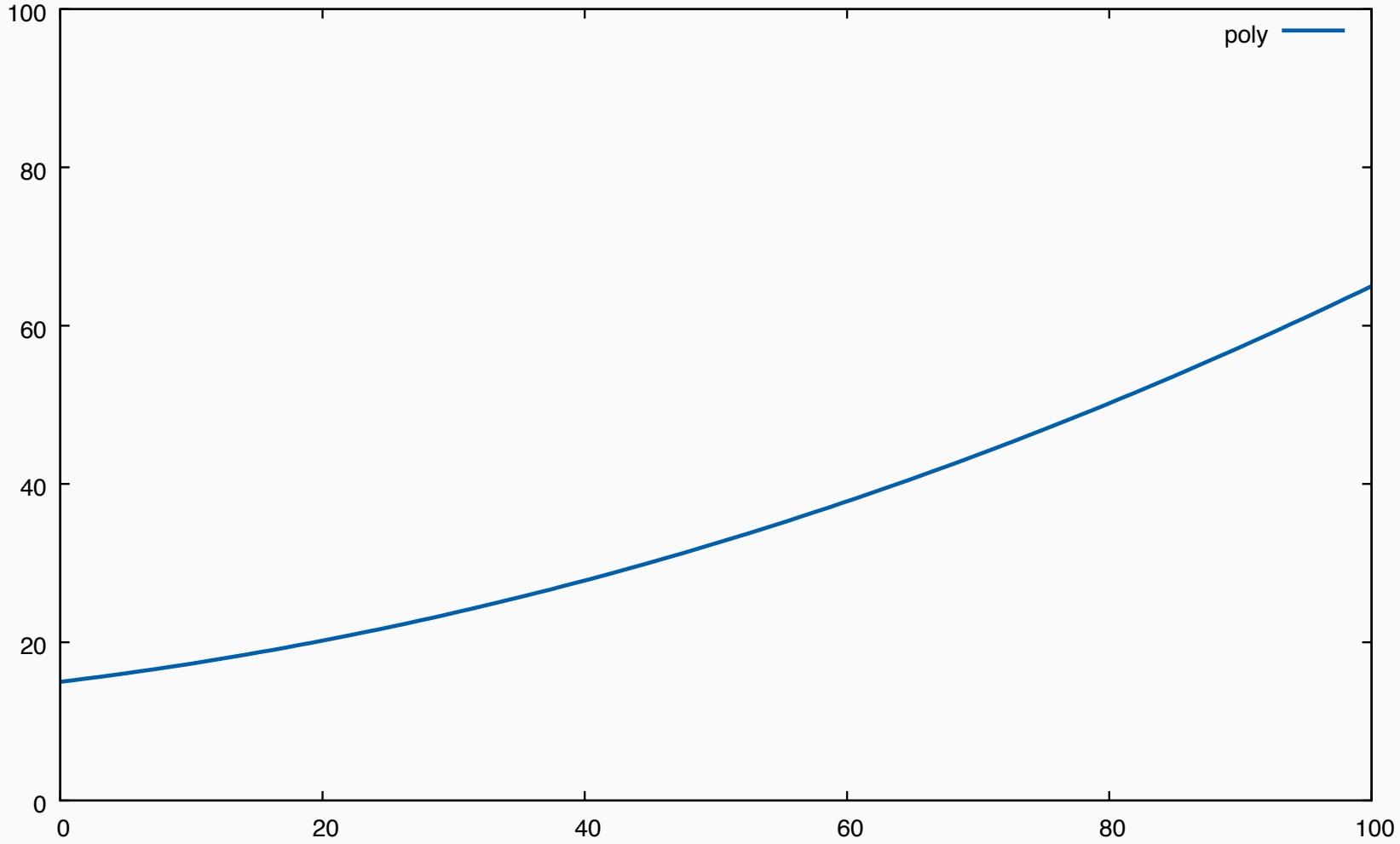
Complexité constante



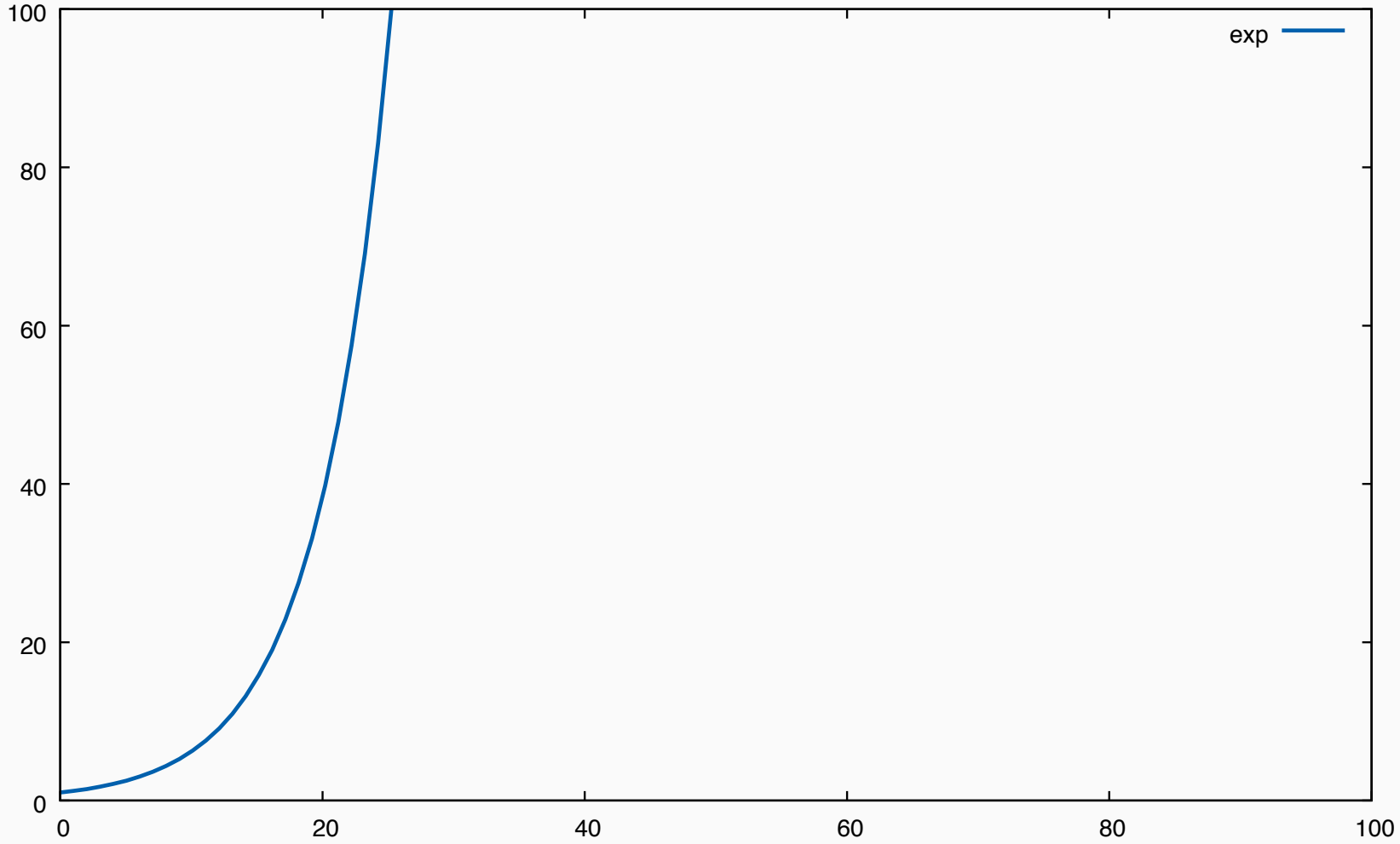
Complexité linéaire



Complexité polynomiale



Complexité exponentielle



Conclusion

- la performance d'un programme n'est pas le premier critère de qualité
- mais peut avoir une grande importance
- estimation en ordre de grandeur

Questions