Mesure de complexité

Necessité

un programme trop lent

est

inutilisable

Mesure de complexité

Necessité

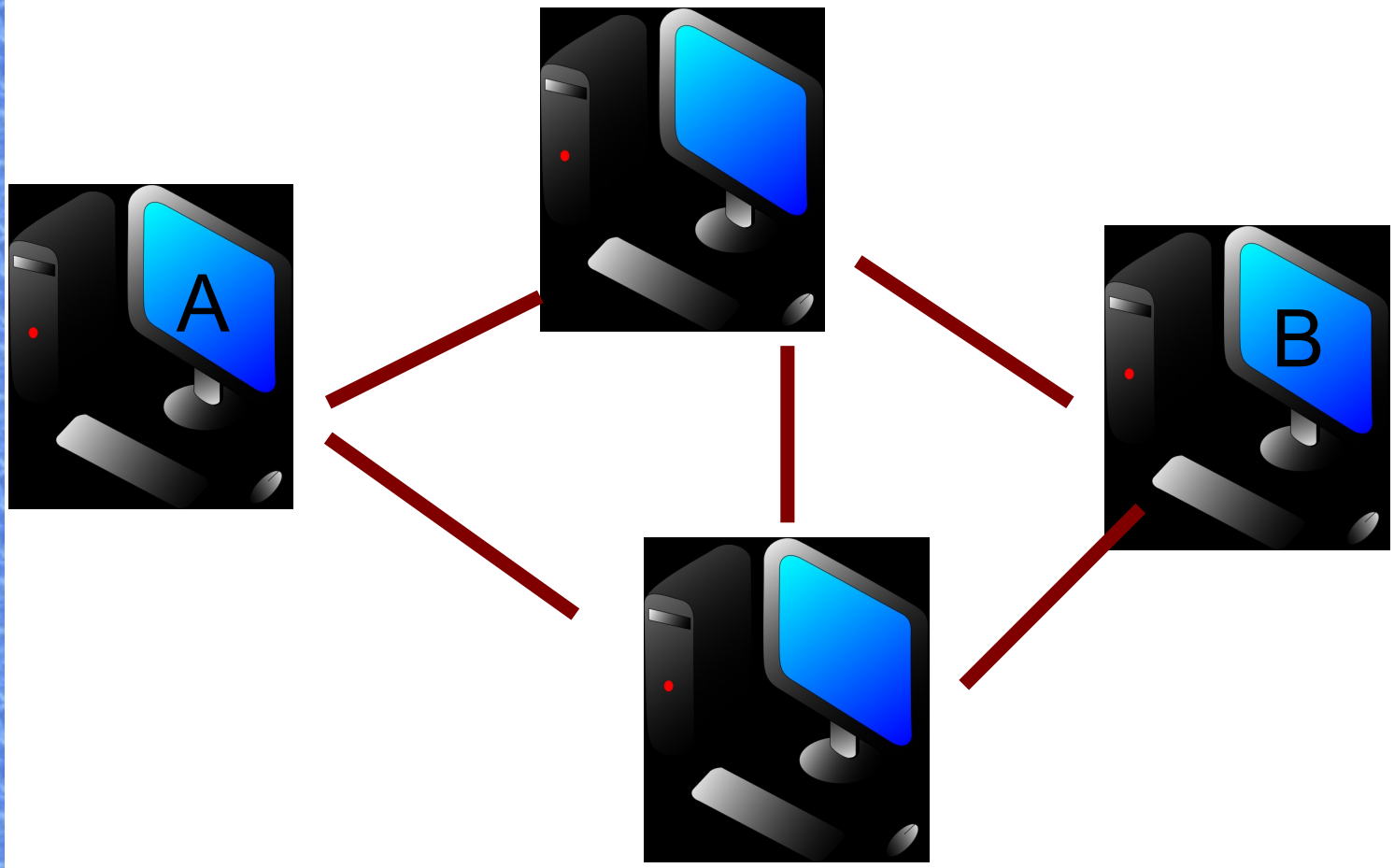
- limite théorique :
  - durée en siècles
- limite pratique :
  - durée en jours
- limite psychologique :
  - plus de 20 s

Mesure de complexité

Necessité

Nombre de chemins possibles d'un point A à un autre B dans un réseau :

4 points .... 4 chemins

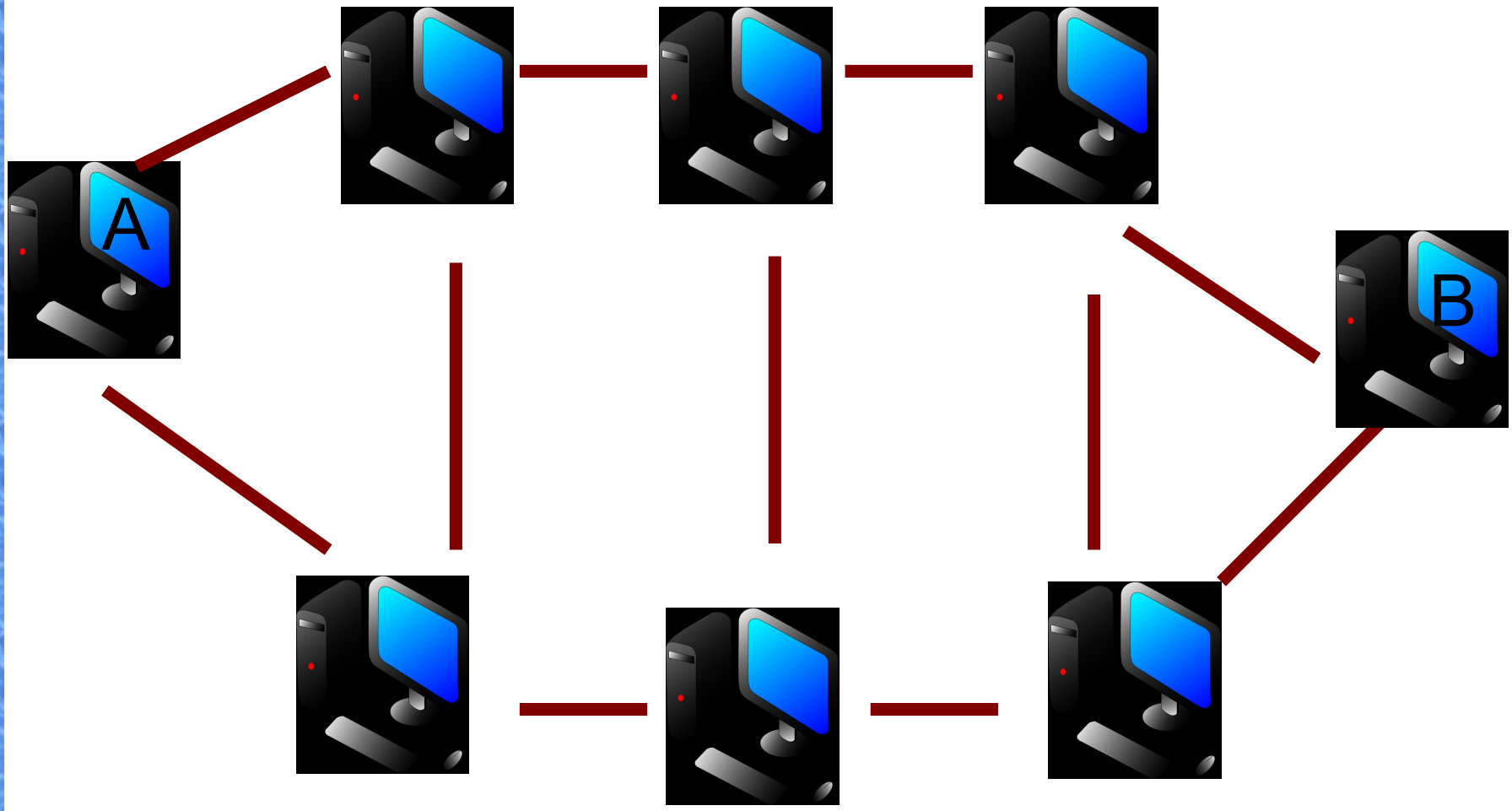


Mesure de complexité

Necessité

Nombre de chemins possibles d'un point A à un autre B dans un réseau :

8 points .... 16 chemins

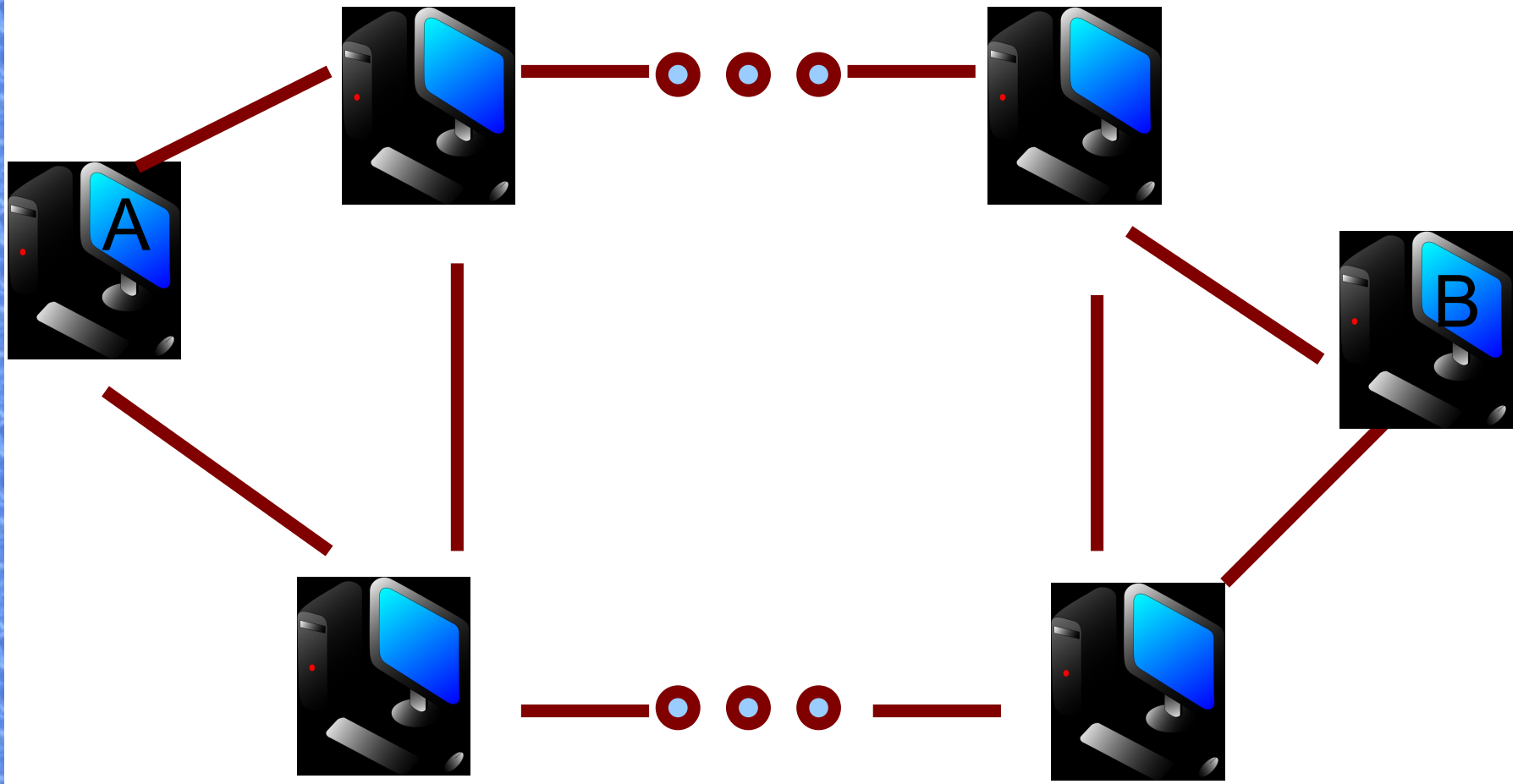


Mesure de complexité

Necessité

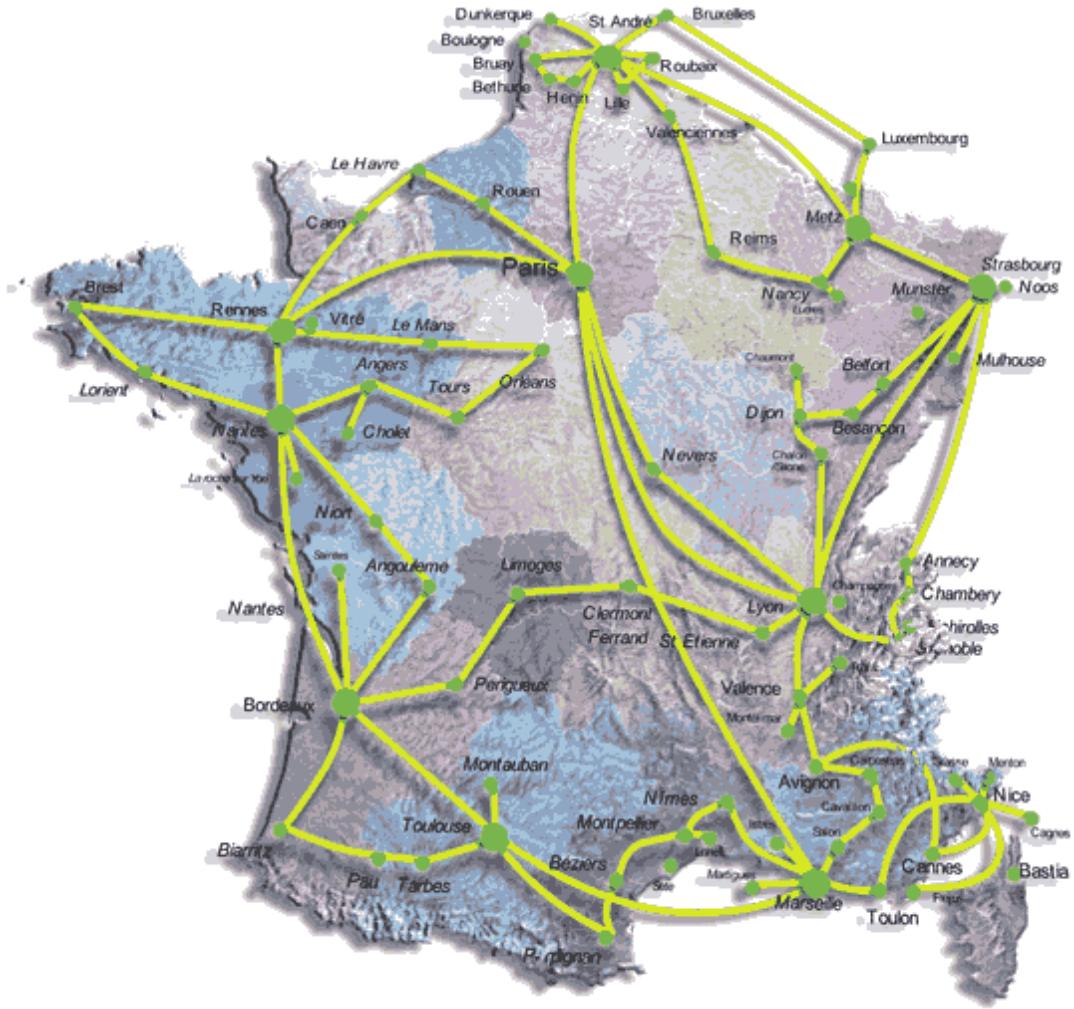
Nombre de chemins possibles d'un point A à un autre B dans un réseau :

10 points .... 32 chemins, 20 points ...



Mesure de complexité
Necessité

réseau Numéricable en 2007



Mesure de complexité

Necessité

Nombre de chemins possibles d'un point A à un autre B dans un réseau :

10 points .... 32 chemins,

20 points ... 1024 chemins,

100 points ... 20^50 chemins (soit 10^15)

Si l'étude d'un chemin nécessite 1 ms, il faut

32 ms pour 10 points,
1 s pour 20 points,

et ...

Mesure de complexité

Necessité

Nombre de chemins possibles d'un point A à un autre B dans un réseau :

10 points .... 32 chemins,

20 points ... 1024 chemins,

100 points ... 2^50 chemins (soit 10^15)

Si l'étude d'un chemin nécessite 1 ms, il faut

32 ms pour 10 points,
1 s pour 20 points,

et pour 100 points, 10^15 ms, soit 10^12 s,

soit 3.10^8 h, soit 12.10^ jours, soit 36.10^3 ans,

soit 360 siècles

Mesure de complexité
Méthodes

- mesurer le temps d'exécution sur de petites valeurs et extrapoler

Mesure de complexité
Méthodes

- mesurer le temps d'exécution sur de petites valeurs et extrapoler
  - précision des mesures
  - qualité de l'extrapolation

Mesure de complexité
Méthodes

- mesurer le temps et extrapoler
- ou prévoir théoriquement le temps

Mesure de complexité
Méthodes

- mesurer le temps et extrapoler
- ou prévoir théoriquement le temps
  - le temps dépend de la machine
  - le temps est partagé
  - le temps est compliqué

Mesure de complexité
Méthodes

- mesurer le temps et extrapoler
- prévoir théoriquement le temps
- mesurer une unité plus simple

Mesure de complexité
Méthodes

- mesurer le temps et extrapoler
- prévoir théoriquement le temps
- mesurer une unité plus simple
  - s'abstraire des machines --> mesurer l'algo
  - simplifier --> unité élémentaire = t

Mesure de complexité
Règles

1) les instructions élémentaires

et les opérations élémentaires coûtent 1 t

Affectation : x ← 12

Opérateurs : + - * / et ou non < ≠

Mesure de complexité
Règles

1) les instructions élémentaires coûtent 1 t

2) séquence --> les coûts s'ajoutent

```
X ← 12; Y ← X*2
```

Mesure de complexité
Règles

1) les instructions élémentaires coûtent 1 t

2) séquence --> les coûts s'ajoutent

```
Coût(A;B)=Coût(A)+Coût(B)
```

Mesure de complexité
Règles

1) les instructions élémentaires coûtent 1 t

2) séquence --> les coûts s'ajoutent

3) conditionnelle --> entre le min et le max

```
si x<13
  alors x ← y+1
  sinon x ← 3*y-5
finsi
```

Mesure de complexité
Règles

1) les instructions élémentaires coûtent 1 t
2) séquence --> les coûts s'ajoutent
3) conditionnelle --> entre le min et le max

Coût**(C)+Min(**Coût**(A),**Coût**(B))**

≤ Coût**(si C alors A sinon B finsi)**

≤ Coût**(C)+Max(**Coût**(A),**Coût**(B))**

Mesure de complexité
Règles

4) répétitives --> somme des coûts

$$Coût(\text{répéter } A \text{ jusqu'à } C) = \sum_{k=1}^{\text{nb de tours}} \left(Coût(A_k) + Coût(C)\right)$$

Mesure de complexité
Règles

4) répétitives --> somme des coûts

$$Coût(\text{tant que } C \text{ faire } A \text{ fin tant que}) =$$

$$Coût(C) + \sum_{k=1}^{\text{nb de tours}} \left(Coût(A_k) + Coût(C) \right)$$

Mesure de complexité
Règles

1) les instructions élémentaires coûtent 1 t
2) séquence --> les coûts s'ajoutent
3) conditionnelle --> entre le min et le max

4) répétitive --> somme des coûts

$$Coût(\text{tant que } C \text{ faire } A \text{ fin tant que}) =$$

$$Coût(C) + \sum_{k=1}^{\text{nb de tours}} \left(Coût(A_k) + Coût(C)\right)$$

Mesure de complexité
Règles

5) les autres structures se traduisent :

```
pour i de valdeb à valfin faire
   A
fin pour
```

équivaut à

```
ifin ← valfin
i ← valfin
tant que i ≤ ifin faire
   A
   i ← i + 1
fin tant que
```

Mesure de complexité
Exemple

```
fonction moitié(n entier):entier
début
   reste ← 0
   tant que reste<n faire
      reste ← reste + 1
      n ← n -1
   fin tant que
   retourner(n)
fin
```

Mesure de complexité
Exemple

```
fonction puissance(n,e entier):entier
début
   result ← 1
   pour cpt de 1 à e faire
      result ← result * n
   fin pour
   retourner(result)
fin
```