

Cours 8 : algorithmes de tris

Jean-Stéphane Varré

Université Lille 1

jean-stephane.varre@lil.fr

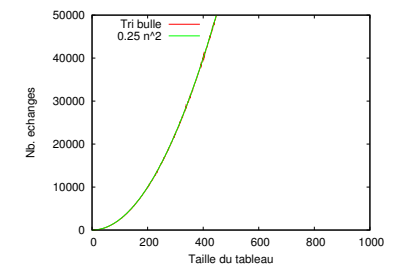
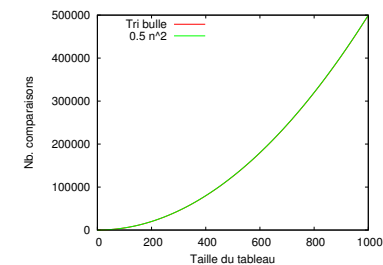
Au menu

- Tri bulle
- Tri insertion
- Tri tas (HeapSort)
- Tri fusion (MergeSort)
- Tri rapide (QuickSort)
- Quelle est la complexité du meilleur algorithme de tri ?
- Trier en $\mathcal{O}(n)$, c'est possible !?

Implémentation

```
procedure triBulle (var t : TABLEAU; var nbcmp : CARDINAL; var nbech : CARDINAL);
var
  i, j : CARDINAL;
begin
  for i := high(t) downto low(t) + 1 do begin
    for j := low(t) to i-1 do begin
      inc(nbcmp);
      if t[j] > t[j+1] then begin
        inc(nbech);
        echanger(t,j,j+1);
      end {if};
    end {for};
  end {for};
end {triBulle};
```

Temps de calcul - tri bulle



Implémentation

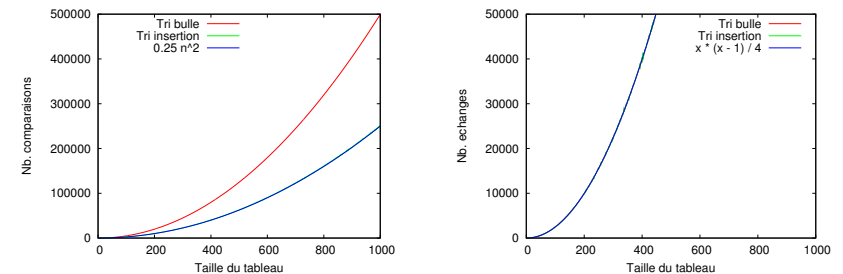
```

procedure inserer (var t : TABLEAU; const n : CARDINAL; var nbcmp : CARDINAL; var
var
  i : CARDINAL;
begin
  i := n;
  while (i > low(t)) and (t[i-1] > t[i]) do begin
    inc(nbcmp);
    inc(nbech);
    echanger(t,i,i-1);
    dec(i);
  end {while};
  if (i > low(t)) and (t[i-1] <= t[i]) then inc(nbcmp);
end {inserer};

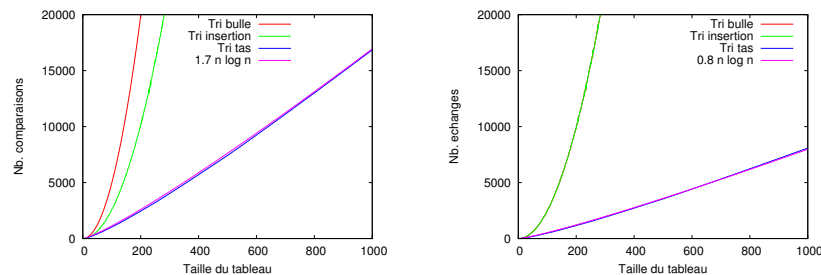
procedure triInsertion (var t : TABLEAU; var nbcmp : CARDINAL; var nbech : CARDI
var
  i : CARDINAL;
begin
  for i := low(t) + 1 to high(t) do begin
    inserer (t,i,nbcmp,nbech);
  end {for};
end {trier};

```

Temps de calcul - tri insertion



Temps de calcul - tri tas



Implémentation

```

procedure triFusion (var t : TABLEAU; const g, d : INTEGER; var nbcmp : CARDINAL;
var
  m : INTEGER;
begin
  if g < d then begin
    m := (g + d) div 2;
    // tri récursif des parties gauche et droite du tableau
    triFusion (t, g, m, nbcmp, nbech);
    triFusion (t, m + 1, d, nbcmp, nbech);
    // fusion des parties triées
    fusionner(t,g,d,m);
  end {if};
end {triFusion};

```

L'appel initial est : `triFusion(t,low(t),high(t),cmp,ech);`

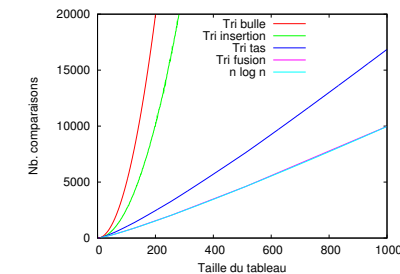
Implémentation

```

procedure fusionner (var t : TABLEAU; const g,d,m : INTEGER);
var
  u : TABLEAU;
  i, j, k : INTEGER;
begin
  setlength(u,length(t));
  for i := m downto g do u[i] := t[i];
  for j := m+1 to d do u[d+m+1-j] := t[j];
  i := g; j := d;
  for k := g to d do begin
    inc(nbcmp);
    if u[i] < u[j] then begin
      t[k] := u[i];
      i := i + 1;
    end else begin
      t[k] := u[j];
      j := j - 1;
    end {if};
  end {for};
end {fusionner};

```

Temps de calcul - tris fusion



Implémentation

```

procedure QSort (var t : TABLEAU; const g, d: INTEGER; var nbcmp, nbech : CARDINAL);
var
  m : INTEGER; { position du pivot }
  v : ELEMENT; { valeur du pivot }
begin
  if g < d then begin
    v := t[g];
    m := g;
    partitionner(t,v,g,d,m,nbcmp,nbech);
    QSort (t, g, m-1, nbcmp, nbech);
    QSort (t, m+1, d, nbcmp, nbech);
  end {if};
end {Qsort};

```

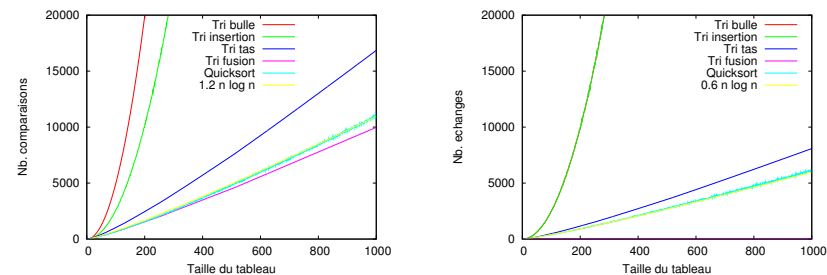
Implémentation

```

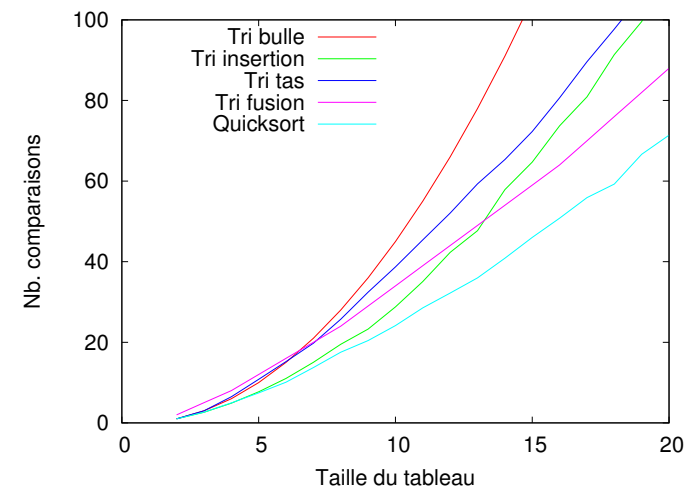
procedure partitionner (var t : TABLEAU;
  const v : ELEMENT; const g, d : CARDINAL; var m : CARDINAL;
  var nbcmp, nbech : CARDINAL);
var
  i, j : CARDINAL;
begin
  for i := g + 1 to d do begin
    inc(nbcmp);
    if t[i] < v then begin
      m := m + 1;
      inc(nbech);
      echanger(t,m,i);
    end {if};
  end {for};
  inc(nbech);
  echanger(t,m,g);
end {partitionner};

```

Temps de calcul - tris rapide



Temps de calcul sur de petites instances



Complexité

	Temps (comparaisons)				Espace
	Meilleur des cas	Pire des cas	Moyenne	Constante mesurée	
Tri bulle	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\frac{1}{2}$	$\Theta(1)$
Tri insertion	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	$\frac{1}{4}$	$\Theta(1)$
Tri par tas	$\Theta(n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$	1.7	$\Theta(1)$
Tri fusion	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	1	$\Theta(n)$
Tri rapide	$\Theta(n \log n)$	$\Theta(n^2)$	$\mathcal{O}(n \log n)$	1.2	$\Theta(1)$

Borne inférieure pour les tris

Théorème
Tout algorithme de tri par comparaisons exige $\Omega(n \log n)$ comparaisons dans le pire des cas.