UNIVERSITÉ D'ORLÉANS **M1 Info**

Informatique Année **2014 - 2015**

Contrôle continu : Programmation Parallèle - 1h

Exercice 1. Parallélisation PRIM (cf exercice 2 TD2 (5pts))

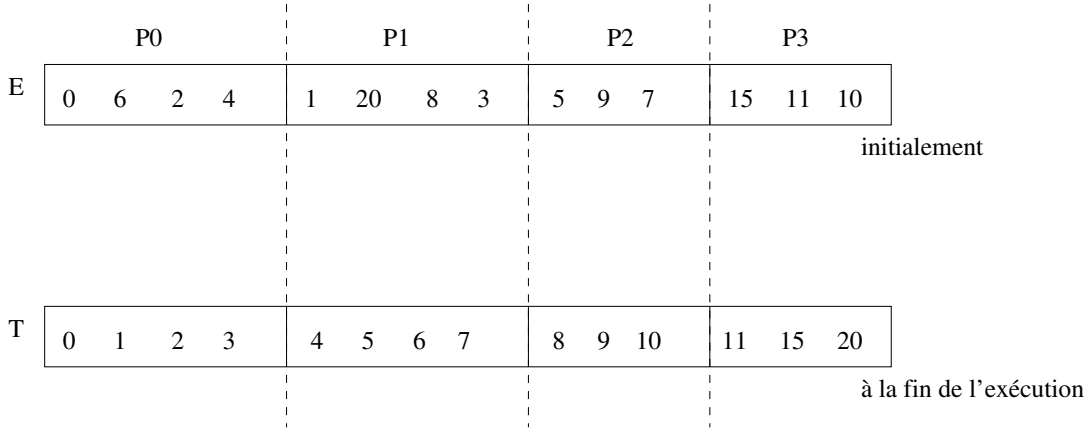
1. Donnez l'ordre de grandeur de la complexité parallèle de l'algorithme PRIM, implémenté en TD, en notant n le nombre de nœuds du graphe étudié et p le nombre de processeurs utilisé.
2. Que se passe-t-il si par hasard les $\frac{n}{p}$ premiers nœuds visités sont gérés par le même processeur ? Quel serait alors le moyen d'améliorer la performance ?

Exercice 2. Parallélisation du tri par induction (15pts)

Soit l'algorithme suivant qui réalise un tri appelé tri par induction

```
for i from 1 to n do
  p = 1
  for j from 1 to n do
    if (E(j)<E(i)) then
      p = p + 1
    end if
  end for
  T(p) = E(i)
end for
```

Soit E un tableau d'entiers tous distincts, on souhaite paralléliser la première boucle for (indice i) en répartissant le calcul des positions p sur nprocs processeurs. Initialement le tableau E est réparti sur les nprocs processeurs. De plus à la fin du tri, on souhaite avoir le tableau T également réparti sur les nprocs processeurs :



1. (5pts) Expliquez les étapes de la parallélisation sachant que le tableau E est déjà distribué sur les processeurs. Vous pouvez faire des schémas pour expliquer en particulier les échanges qui seront nécessaires.
2. (5pts) Par rapport à ces étapes, explicitez les fonctions de communications MPI que vous allez utiliser et pourquoi.
3. (5pts) A partir de la trame ci-dessous proposez une implémentation MPI. Pour certains calculs (des indices par exemple), vous pouvez utiliser une fonction que vous définirez clairement sans l'implémenter.

```
int main ( int argc , char **argv ) {
  int pid, nprocs;
  MPI_Init (&argc , &argv) ;
  MPI_Comm_rank(MPI_COMM_WORLD, &pid ) ;
  MPI_Comm_size (MPI_COMM_WORLD, &nprocs ) ;

  int n_local = atoi(argv[1]);
  int root = atoi(argv[2]);

  srand(time(NULL)+pid);
  int* E_local = new int[n_local];
  generation_elts_distincts(E_local,n_local);

  delete[] E_local;
  MPI_Finalize() ;
  return 0 ;
}
```