Qüestió 1 (1.1 punts)

Donat el següent codi principal d'un programa, on suposem que `n` és una constant definida amb un valor enter:

```
float a;
float A[n][n], X[n][n], Y[n][n], Z[n][n];
T1( A, X, Y, Z );
a = T2( A );
T3( a, X );
T4( a, Y );
T5( A, A, Z );
T6( X, Y, Z );
```

i donat el codi de les següents funcions:

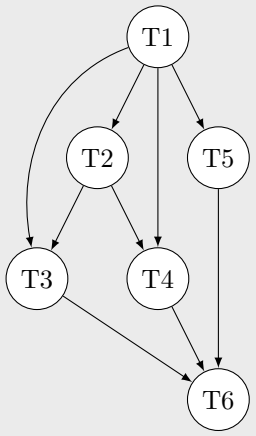
```
float T2( float A[n][n] ) {
  float a = 0.0;
  for( int i=0; i<n; i++ )
    for( int j=i; j<n; j++ )
      for( int k=i; k<n; k++ )
        a = a + A[i][k];
  return a;
}

void T5( float X[n][n], float Y[n][n], float Z[n][n] ) {
  float aux;
  for( int i=0; i<n; i++ )
    for( int j=0; j<n; j++ ) {
      aux=0;
      for ( int k=0; k<n; k++ )
        aux += X[i][k]*Y[k][j];
      Z[i][j]=aux;
    }
}
```

on sabem que la funció `T1` modifica tots els seus arguments i té un cost de n^3 flops, mentre que les funcions `T3`, `T4` i `T6` modifiquen només el seu últim argument i tenen també un cost de n^3 flops. La funció `T2` té un cost de $\frac{n^3}{3}$ i la funció `T5` té un cost de $2n^3$.

0.3 p. (a) Dibuixa el graf de dependències de dades entre les tasques.

Solució:



0.6 p. (b) Implementa una versió paral·lela mitjançant OpenMP utilitzant una sola regió paral·lela.

Solució:

```
float a;
float A[n][n], X[n][n], Y[n][n], Z[n][n];
T1( A, X, Y, Z );
#pragma omp parallel
{
  #pragma omp sections
  {
    #pragma omp section
    a = T2( A );
    #pragma omp section
    T5( A, A, Z );
  }
  #pragma omp sections
  {
    #pragma omp section
    T3( a, X );
    #pragma omp section
    T4( a, Y );
  }
} /* Fi del parallel */
T6( X, Y, Z );
```

0.2 p. (c) Obteniu l'speedup de la versió paral·lela de l'apartat anterior per a 2 processadors.

Solució:

Temps d'execució seqüencial:

$$t(n) = n^3 + \frac{n^3}{3} + n^3 + n^3 + 2n^3 + n^3 = \left(6 + \frac{1}{3}\right) n^3 = \frac{19}{3} n^3 \text{ flops}$$

Temps d'execució paral·lel per a $p = 2$:

$$t(n,p) = n^3 + \max\left(\frac{n^3}{3}, 2n^3\right) + \max(n^3, n^3) + n^3 = 5n^3 \text{ flops}$$

Speedup:

$$S = \frac{\frac{19}{3}n^3}{5n^3} = \frac{19}{15} = 1.27$$

Qüestió 2 (1.2 punts)

Es vol gestionar un concurs de `nP` participants, per part d'un jurat compost per `nM` membres, numerats des de la posició 0 en avant. Cada jurat emet dos vots, un de 10 punts per a un participant i un altre de 5 per a un altre. Per a això, la funció `gestiona_vots` rep el vector `vots_jurat`, de `nM`*2 components, amb els dos vots emesos per cada integrant del jurat i completa el vector `pts_participants`, de `nP` elements, amb la puntuació aconseguida per cada participant. A més de calcular quin ha sigut el participant que ha guanyat el concurs (màxima puntuació), la funció completa els vectors `m10pts` i `m5pts` amb els identificadors dels membres del jurat que han concedit 10 punts o 5 punts, respectivament, al participant indicat com a argument de la funció. Al final, obté en la variable `nPZeroPts` el nombre de participants sense cap vot.

```
void gestiona_vots(int vots_jurat[], int participant) {
  int i, p10pts, p5pts, pts_participants[nP];
  int m10pts[nM], m5pts[nM];
  int nM10pts=0, nM5pts=0, nPZeroPts=0;
  int pts_max=0, guanyador;
  ...
  for (i=0;i<nM;i++) {
    // Acumulem els punts als participants
    p10pts=vots_jurat[i*2];
    p5pts=vots_jurat[i*2+1];
    pts_participants[p10pts]+=10;
    pts_participants[p5pts]+=5;
    // Obtenim els membres que han votat al participant indicat
    if (p10pts==participant) {
      m10pts[nM10pts]=i;
      nM10pts++;
    }
    else if (p5pts==participant) {
      m5pts[nM5pts]=i;
      nM5pts++;
    }
  }
  // Calculem el guanyador i el nombre de participants amb 0 punts
  for (i=0;i<nP;i++) {
    if (pts_participants[i]>pts_max) {
      pts_max=pts_participants[i];
      guanyador=i;
    }
    else if (pts_participants[i]==0)
      nPZeroPts++;
  }
  ...
}
```

Paral·lelitzeu `gestiona_vots` de la forma més eficient possible, emprant una única regió paral·lela.

Solució:

```
void gestiona_vots(int vots_jurat[], int participant) {
  int i, p10pts, p5pts, pts_participants[nP];
  int m10pts[nM], m5pts[nM];
  int nM10pts=0, nM5pts=0, nPZeroPts=0;
  int pts_max=0, guanyador;
  ...
  #pragma omp parallel
  {
    #pragma omp for private(p10pts, p5pts)
    for (i=0;i<nM;i++) {
      // Acumulem els punts als participants
      p10pts=vots_jurat[i*2];
      p5pts=vots_jurat[i*2+1];
      #pragma omp atomic
      pts_participants[p10pts]+=10;
      #pragma omp atomic
      pts_participants[p5pts]+=5;
      // Obtenim els membres que han votat al participant indicat
      if (p10pts==participant) {
        #pragma omp critical (pts10)
        {
          m10pts[nM10pts]=i;
          nM10pts++;
        }
      }
      else if (p5pts==participant) {
        #pragma omp critical (pts5)
        {
          m5pts[nM5pts]=i;
          nM5pts++;
        }
      }
    }
    // Calculem el guanyador i el nombre de participants amb 0 punts
    #pragma omp for reduction(+:nPZeroPts)
    for (i=0;i<nP;i++) {
      if (pts_participants[i]>pts_max) {
        #pragma omp critical
        if (pts_participants[i]>pts_max) {
          pts_max=pts_participants[i];
          guanyador=i;
        }
      }
      else if (pts_participants[i]==0)
        nPZeroPts++;
    }
  }
  ...
}
```

Qüestió 3 (1.2 punts)

Donada la següent funció:

```
void normalitza_mat(double A[N][N]){
  int i, j;
  double s, norm1=0, norm2=0;
  for (i = 0; i < N; i++){
      s = 0;
      for (j=0; j<N; j++){
          norm1+=A[i][j];
          s+= fabs(A[i][j]);
      }
      if (s>norm2)
         norm2= s;
  }
  norm1=sqrt(norm1)*norm2;
  for (i = 0; i < N; i++)
      for (j = 0; j < N; j++)
          A[i][j]*=norm1;
}
```

0.9 p. (a) Paral·lelitza mitjançant OpenMP la funció anterior, usant per a això una sola regió paral·lela.

Solució:

```
#pragma omp parallel
{
  #pragma omp for private(j,s) reduction(+:norm1) reduction(max:norm2)
   for (i = 0; i < N; i++){
       ...
   }
  #pragma omp single
  norm1=sqrt(norm1)*norm2;
  #pragma omp for private(j)
   for (i = 0; i < N; i++)
       for (j = 0; j < N; j++)
           A[i][j]*=norm1;
}
```

0.2 p. (b) Calcula el cost seqüencial i paral·lel, suposant que N és divisible entre el nombre de fils p i que el cost de les funcions `fabs` i `sqrt` és d'1 flop. Indicar els càlculs intermedis per a aconseguir la solució.

Solució: Cost seqüencial:

$$t(N) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} 3 + 2 + \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} 1 = 3N^2 + 2 + N^2 \approx 4N^2 flops.$$

Cost paral·lel:

$$t(N,p) = \sum_{i=0}^{N/p-1}\sum_{j=0}^{N-1} 3 + 2 + \sum_{i=0}^{N/p-1}\sum_{j=0}^{N-1} 1 = \frac{3N^2}{p} + 2 + \frac{N^2}{p} \approx \frac{4N^2}{p} flops.$$

0.1 p. (c) Calcula l'speedup i l'eficiència.

Solució:

$$S(n,p) = \frac{4N^2}{\frac{4N^2}{p}} = p.$$

$$E(n,p) = \frac{S(n,p)}{p} = 1.$$