Considere que Ud. viaja a Europa y puede llevar una maleta de hasta *maxW* kilos. Dispone de un conjunto de *n* artículos $\{A_0, A_1, ..., A_{n-1}\}$. El artículo A_i pesa *w[i]* kilos y vale *v[i]* euros. No puede llevar todos los artículos porque la suma de sus pesos excede *maxW*. Debe elegir qué artículos llevar maximizando la suma de sus valores. Este problema se conoce como Knapsack 0-1 y está en la categoría *NP-difícil*. El mejor algoritmo conocido que calcula la solución óptima toma demasiado tiempo: es $O(2^n)$. La función *llenarMaleta* de abajo es una heurística: entrega una buena solución para este problema y en un tiempo razonable, pero no es la solución óptima. Para lograrlo genera aleatoriamente *k* subconjuntos de artículos y elige el de mayor valor que no exceda *maxW*. Al retornar, la solución del problema se entrega en el parámetro de salida *z*: el subconjunto con los artículos elegidos es $\{ A_i \mid tq\ \mathbf{z[i]}=1\}$. La función *random0or1* entrega aleatoriamente 0 o 1.

```
double llenarMaletaSec(double w[], double v[], int z[], int n,
                       double maxW, int k) {
  double best= -1;
  while (k--) {
    int x[n];
    double sumW= 0, sumV= 0;
    for (int i=0; i<n; i++) {
      x[i]= random0or1() && sumW+w[i]<=maxW ? 1 : 0;
      if (x[i]==1) {
        sumW += w[i];
        sumV += v[i];
    } }
    if (sumV>best) {
      best= sumV;
      for(int i=0; i<n; i++) {
        z[i]= x[i];
  } } }
  return best;
}
```

Programe la función *llenarMaletaPar* usando la misma heurística pero de modo que la elección se haga en paralelo para una máquina con 8 cores. Esta función recibe los mismos parámetros que *llenarMaletaSec*.

Metodología obligatoria: Lance 8 nuevos threads invocando 8 veces *pthread_create*. Cada thread evalúa *k/8* subconjuntos aleatorios invocando *llenarMaletaSec(..., k/8)*. Defina una estructura con campos para todos los parámetros que recibirá *llenarMaletaSec* y otro campo para el valor retornado. Cuidado: no puede compartir el arreglo *z* entre todos los threads. Debe crear un arreglo *z* independiente para cada thread. Use el thread principal solo para crear los threads y para elegir la mejor solución entre las mejores encontradas por los 8 threads. Si la mejor solución fue por ejemplo la del thread 3, *llenarMaletaPar* debe retornar el valor *best* que calculó el thread 3 y copiar el arreglo *z* calculado por el thread 3 en el parámetro *z* de *llenarMaletaPar*. La forma de crear los threads es muy similar a la manera en que se crearon los threads para resolver en paralelo el problema de la búsqueda de un factor en la clase auxiliar del viernes 6 de noviembre.

Se requiere que el incremento de velocidad (*speed up*) sea al menos un factor 1.7x. Cuando pruebe su tarea en su notebook asegúrese de que posea al menos 2 cores, que esté configurado en modo alto rendimiento y que no estén corriendo otros procesos intensivos en uso de CPU al mismo tiempo. De otro modo podría no lograr el *speed up* solicitado.

Instrucciones

Baje *t5.zip* de U-cursos y descomprímalo. El directorio *T5* contiene los archivos *test-t5.c*, *Makefile* y *maleta.h* (con los encabezado requeridos). Ud. debe crear el archivo *t5.c* y programar ahí la función *llenarMaletaPar*. No olvide el *#include "maleta.h"*.

Pruebe su tarea bajo Debian 10 (con soporte para programas de 32 y 64 bits) con los comandos:

- *make test-g*
- *make test-O*
- *make test-O-m32*
- *make test-valgrind*

Ud. necesita cumplir el requisito de un speed up de 1.7x solo al ejecutar con *make test-O*. No se preocupe si en las otras ejecuciones no cumple ese requisito. La ejecución con *make test-valgrind* no debe reportar errores de manejo de memoria o fugas de memoria. Se descontará medio punto si alguna de las compilaciones reporta algún warning. Copie la salida de todos estos comandos y péguela en el archivo *resultados.txt*.

Entrega

Ud. solo debe entregar los archivos *t5.c* y *resultados.txt* en el formato *.zip* por medio de U-cursos. Se descontará medio punto por día de atraso. No se consideran los días de vacaciones, sábado, domingo o festivos.