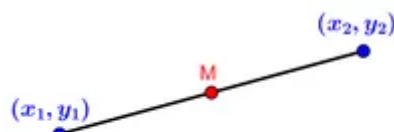


PARTE SÍNCRONA

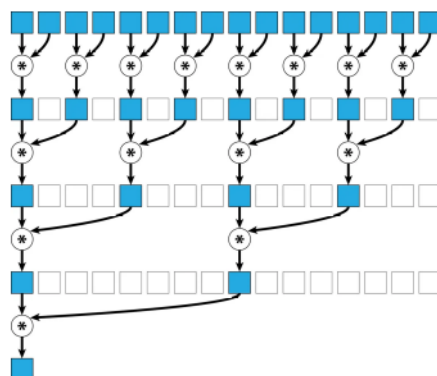
Utilizando la siguiente [plantilla.py](#) Se pide:

- a) Desarrollar una función que dado dos puntos XYZ de tipo `numpy.ndarray` pueda obtener el punto medio de ellos. Puede guiarse por la siguiente fórmula 2D para deducir el caso para 3 dimensiones. **(0.5 puntos)**

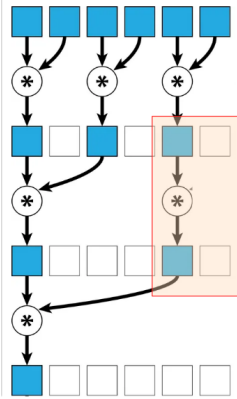

$$M = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$$

- b) Tomando como referencia la función que desarrollo en el inciso “a”. Se pide desarrollar una función que tenga como entrada un arreglo de N puntos y pueda hallar el punto medio total. Vea la siguiente figura, en donde se tienen 16 puntos, de los cuales se obtienen 8 puntos medios para cada par, luego se repite el proceso para obtener 4 puntos medios de los 8 puntos previos, y así hasta finalmente obtener un punto medio total. **(2 puntos)**

Nota: Para este inciso puede considerar que N es un número potencia de 2



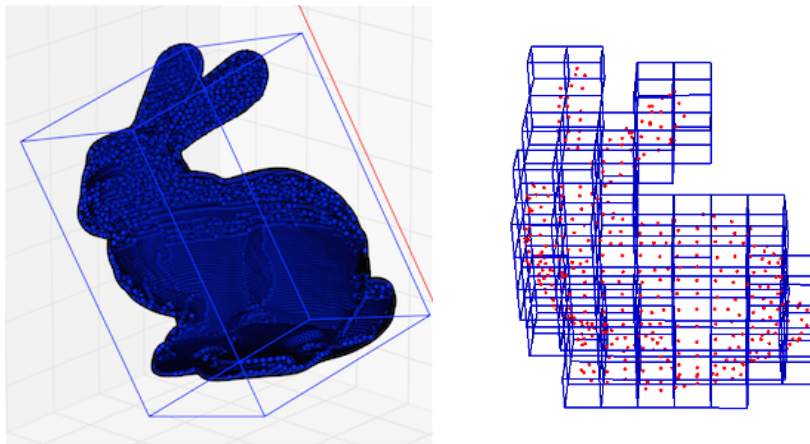
- c) Para este inciso, la función anterior debe obtener el punto medio total para cualquier valor de N. Tenga lo siguiente en cuenta: si el último punto del arreglo no tiene pareja, el punto medio será el mismo número. Ejemplo, para el caso de un arreglo de 6 puntos, se obtendrían 3 puntos medios, en esta etapa sólo se puede formar una única pareja mientras que el otro número se considerará como punto medio, finalmente se obtiene una única pareja para obtener el punto final (vea la siguiente figura como referencia) **(4 puntos)**



- d) Aplicar Multiprocesing para 7 distintos arreglos con N=777 cantidad de puntos. Imprima el identificador del proceso y los tiempos de ejecución para cada caso. **(1.5 puntos)**
- e) ¿Los procesos terminaron en orden cronológico (es decir, proceso1 luego proceso2,...)? Comente si hay coherencia (o no) en el resultado obtenido **(1 puntos)**
- f) ¿Qué diferencia existe entre Multiprocessing y Threading? ¿En que escenarios conviene aplicar cada método? **(1 punto)**

Alternativa: (Solo puede reemplazar la nota de los incisos d,e,f)

Lo que ha desarrollado es parte de las etapas para realizar un *voxel downsampling* a una nube de puntos. Entiéndase como nube de puntos como puntos definidos en el espacio que representan un objeto o zona (Vea la siguiente figura que muestra el ejemplo de la nube de puntos de un conejo). Y, entiéndase como voxel un espacio cuboide que almacena cierta cantidad de puntos



Para realizar un voxel downsampling primero es necesario dividir la nube de puntos en distintos voxels y luego para cada voxel que encierra un grupo de puntos se obtiene un único punto. De esta forma se reduce la cantidad de puntos. Se pide:

- Utilizando la plantilla [BunnyVoxelDownsampling.py](#) usted debe adaptar la función que desarrollo en el inciso c) para poder aplicar el Voxel-Downsampling a la nube de puntos [StanfordBunny.csv](#) **(1.5 puntos)**
- Aplicar Multiprocesing para los distintos valores de *numDivs*: [10,30,80,150,300], imprima los tiempos de ejecución. Realice las modificaciones necesarias **(2 puntos)**

PARTE ASÍNCRONA

Pregunta 1 (4 puntos)

Dado el polinomio:

$$f(x) = x + 2x^2 + 3x^3 + 4x^4 + \dots + 10000x^{10000}$$

Calcular $f(2022)$. Para ello:

- (1 punto)** Escriba una función f cuyo parámetro de entrada sea X y retorne el valor de $f(x)$. Este cálculo debe hacerse en serie, es decir no use multiprocessing. Calcule el tiempo de ejecución para $f(2022)$
- (3 puntos)** Escriba la versión en paralelo de f , de tal manera de que el cálculo de $f(x)$ sea paralelo en 4 procesos y el cálculo se haga más rápido. Calcule el Speed up respecto a la parte a)

Nota: Al final de su archivo en Python, debe verificar que el resultado de la parte a) sea igual a la parte b). Para ello agregue la siguiente línea al final de su programa:

```
assert resultado_serial == resultado_paralelo
```

Dónde `resultado_serial` y `resultado_paralelo` son las variables de su programa que contienen los resultados de las partes a) y b). Si los resultados son iguales, la sentencia `assert` permitirá que su programa termine exitosamente; caso contrario le generará un error, y eso será evidencia de que sus resultados no coinciden.

Pregunta 2 (6 puntos)

Los archivos [bi_data_voltajes_mayo_todos.csv](#) y [bi_data_voltajes_junio_todos.csv](#) son archivos csv de mediciones de voltajes de medidores trifásicos de los meses de mayo y junio respectivamente. La columna `id` corresponde al identificador de la medición, `meter id` es el identificador del medidor, `timestamp` indica la fecha- hora de la medición, y `V1`, `V2` y `V3` son los valores de voltaje de cada una de las fases.

Se le pide escribir una función que tenga como único parámetro de entrada el nombre de un archivo y responder:

- (3 puntos)** ¿Qué mes tiene mayor cantidad de mediciones (filas del csv) donde al menos una de las fases está en corte? Considerar que una fase está en corte si su voltaje es menor a 50V.
- (3 puntos)** ¿Qué mes tiene mayor cantidad de mediciones (filas del csv) con sobretensión en al menos una de las fases? Considerar sobretensión cuando el voltaje es mayor a 277V.

La función que escriba debe estar en paralelo para procesar ambos archivos csv a la vez. Si los archivos no están siendo procesados en paralelo usando multiprocessing, no se le considerará puntaje.