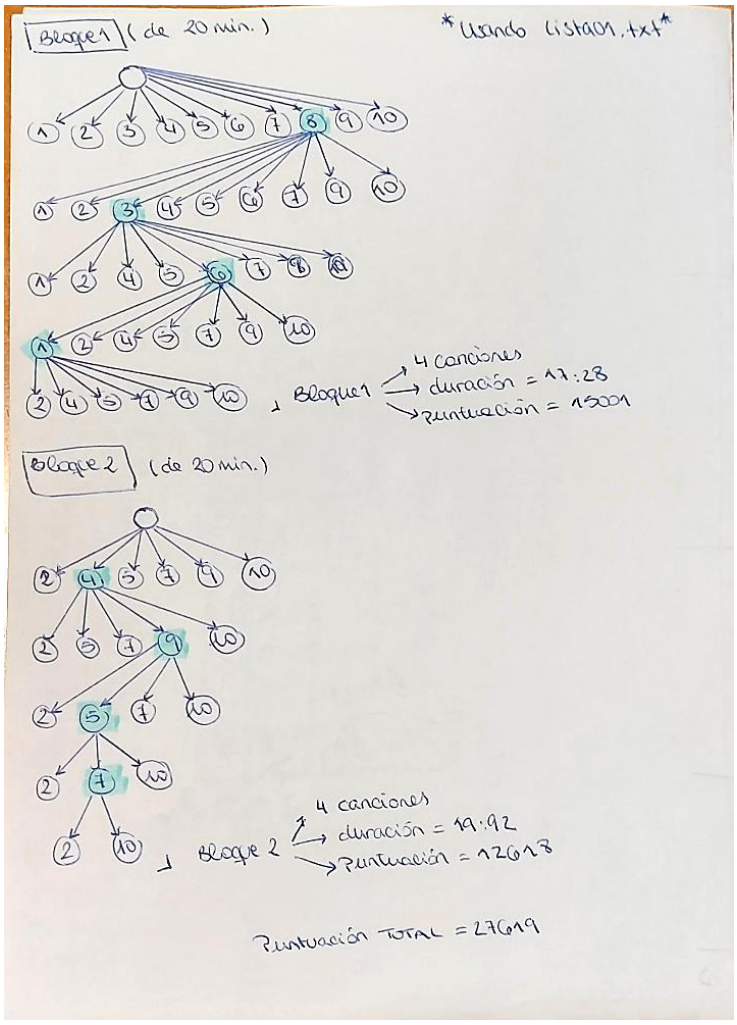
Procesador:	11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.8GHz
Memoria:	16 GB

El algoritmo de backtracking posee una complejidad exponencial, O(n!), ya que el grado del árbol se va reduciendo en cada nivel debido a las comprobaciones de nodo válido.

Ejemplo Lista01.txt y bloques de 20 minutos: Mi árbol comienza con 10 nodos. A medida que se va eligiendo un nodo, en cada nivel se reduce una unidad el grado del árbol.

Proporciona la solución óptima al problema escogiendo, mediante un algoritmo recursivo, el nodo mejor, es decir, el nodo cuya puntuación es la más alta siendo su duración adecuada para poder entrar en el bloque determinado sin necesidad de fragmentarse.

Rellené el bloque 1 y, posteriormente, sin tomar los nodos invalidados rellené el bloque 2.



Salida de la consola de mi clase ‘MejorLista’ para las canciones de la Lista01 y bloques de 20 minutos:

```
A partir del fichero:
C:\Users\mayte\git\alg_FernandezMariaTeresaUO263728/src/main/java/algestudiante/p6/datos/List
a01.txt
Creo la mejor lista con bloques de 20 minutos
Número de canciones: 10
Duración (en min.): 47.3
Puntuación: 33287
***Bloque1:
Cancion [identificador=06rwq3, duracion (en seg.)=288, puntuacion=3842]
Cancion [identificador=0fmvy3, duracion (en seg.)=280, puntuacion=3842]
Cancion [identificador=2lsdf9, duracion (en seg.)=202, puntuacion=3842]
Cancion [identificador=3ld4R7, duracion (en seg.)=267, puntuacion=3475]
Número de canciones: 4
Duración (en min.): 17.283333333333335
Puntuación: 15001
***Bloque2:
Cancion [identificador=8id4R7, duracion (en seg.)=267, puntuacion=3475]
Cancion [identificador=87UKo2, duracion (en seg.)=207, puntuacion=3475]
Cancion [identificador=9u4gE3, duracion (en seg.)=419, puntuacion=2834]
Cancion [identificador=3j4yQ6, duracion (en seg.)=302, puntuacion=2834]
Número de canciones: 4
Duración (en min.): 19.916666666666668
Puntuación: 12618
Puntuacion TOTAL: 27619
```

Mi algoritmo de backtracking, aunque llega a la solución óptima pedida, no debe estar correctamente implementado pues su complejidad es O(n), según la gráfica de la toma de tiempos.

Tabla de tiempos medida con la clase ‘MejorListaTiempos’ sobre canciones aleatorias:

n	tBacktracking	tBacktracking/nVeces
	milisegundos	segundos
5	145	0,0145
10	299	0,0299
15	334	0,0334
20	460	0,046
25	552	0,0552
30	641	0,0641
35	702	0,0702
40	801	0,0801
45	902	0,0902
50	800	0,08
55	814	0,0814
60	878	0,0878
65	1009	0,1009
70	1093	0,1093
75	1167	0,1167
80	1376	0,1376
85	1310	0,131
90	1487	0,1487

95	1505	0,1505
100	1575	0,1575

Teniendo en cuenta que se usó un nVeces = 10000000.

Dichos valores dan lugar a la siguiente gráfica: