Guion 7:

Actividad 1:

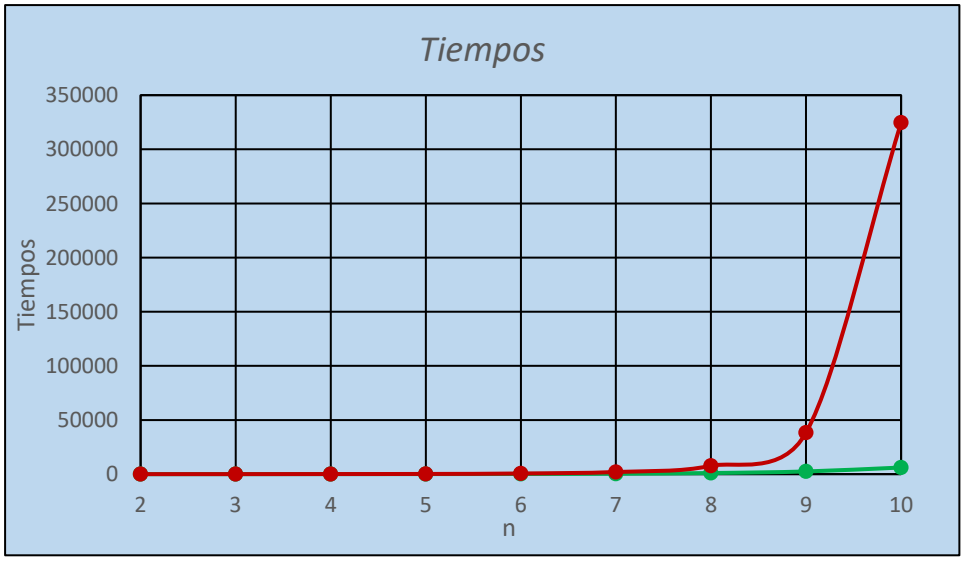
Implementación del código:

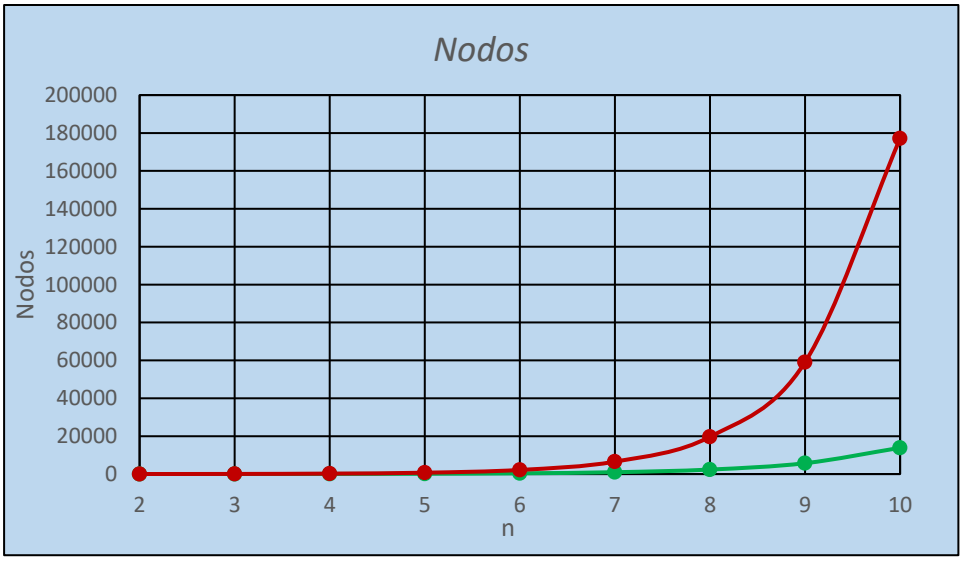
```
TESTING VORAZ:
  -ZNCC: 0,046986
  -Contador: 7
TESTING BACKTRACKING SIN BALANCEO:
  -ZNCC: 0,057968
  -Contador: 3280
TESTING BACKTRACKING CON BALANCEO:
  -ZNCC: 0,053536
  -Contador: 984
TESTING BRANCH AND BOUND:
  -ZNCC: 0,057968
  -Contador: 3280
```

Actividad 2:

Medición de tiempos:

n	t BT balanceo	t BnB	Nodes BT balanceo	Nodes BnB	ZNCC BT balanceo	ZNCC BnB
2	7	9	11	26	0.0	0.0
3	15	32	28	80	0.0314412489533	0.0314412489533
4	39	107	69	242	0.0187037196010	0.0243896357715
5	72	197	168	728	0.0392656438052	0.0392656438052
6	171	607	407	2186	0.0495031699538	0.0522303804755
7	417	1999	984	6560	0.0532466918230	0.0570953637361
8	1024	7719	2377	19682	0.0553396232426	0.0600959062576
9	2496	38350	5740	59048	0.0726382732391	0.0726382732391
10	6144	324698	13859	177146	0.0715564414858	0.0741472840309





Respuesta a las preguntas:

- Compara los resultados proporcionados, número de nodos y tiempos para Backtracking (implementado en la práctica anterior) y R&P.
  Como se puede observar arriba, BranchAndBound tarda mucho más tiempo con respecto a Backtracking con Balanceo, esto se debe a que el heurístico empleado en el BranchAndBound no descarta ningún nodo, lo que hace que tenga que comprobarlos todos, sin embargo, en el caso de Backtracking debido a la condición de balanceo, reduce el número de nodos a comprobar (como se demuestra también arriba). Debido a esto los tiempos de ejecución serán muy superiores en BranchAndBound, y, al visitar todos los nodos, el ZNCC final también será un poco superior, ya que, al no descartar ninguno, se evitan errores, estos resultados coinciden con los de Backtracking sin balanceo.

- Basándote en los resultados anteriores, explica las diferencias de eficiencia entre Backtracking y BnB para resolver este problema. ¿Qué implementación requiere más tiempo de ejecución? ¿Qué implementación genera menos nodos?
  Tal y como se explicó arriba, BranchAndBound requerirá un mayor tiempo de ejecución, justamente porque el número de nodos a explorar es superior con respecto al otro algoritmo, por lo tanto, genera menos nodos Backtracking con Balanceo.