**Informe: Trabajo Practico 2.**

**Tema: Backtracking.**

**Índice:**

**Introducción\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3**

**Desarrollo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4**

**Seguimiento\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5**

**Comparación\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7**

**Conclusión\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8**

**Introducción:**

Para este trabajo practico se debe implementar un programa que obtenga la solución óptima para el problema planteado.

Problema:

Un parque nacional decide abrir sus puertas para permitirle a las familias que puedan ir a conocerlo. Luego de abierta la inscripción, un número de familias se anotan para realizar la visita. Debido a la gran demanda de público se decide establecer un número limitado de días durante los cuales se aceptarán visitas por esta temporada. Sabiendo que el parque tiene una capacidad diaria acotada, los guías deben distribuir a los visitantes durante los días disponibles. Para esto, solicita a cada familia que le informe (en orden de preferencia) los días en los cuales preferirían realizar la visita, y cuántas personas conforman el grupo familiar. Dado que no todas las personas podrán asistir el día que prefieren, el parque decide crear un bono compensatorio para las familias que no son asignadas al día que eligieron como prioritario.

Solución propuesta:

Para la resolución del problema se implementa en c++ un algoritmo de Backtracking. **Backtracking** es una estrategia para encontrar soluciones a problemas que satisfacen restricciones. En este caso se utiliza como restricción el cupo de personas que pueden asistir por día (30 personas).

Se usa un arreglo llamado “personas” en el que se almacena la restricción, otro arreglo de vectores llamado “familias” que contiene el orden en el que se asignan a las familias, y un entero llamado “bono\_obtenido” en el que se almacena el monto obtenido.

**Desarrollo:**

Para el desarrollo de la resolución, se expone el árbol de exploración:

Estado inicial: En el estado inicial podemos encontrar el bono\_obtenido vacío y el arreglo de personas inicializado en 0.

Datos almacenados en un estado: bono\_obtenido (entero)

Personas [10] (arreglo de enteros)

Familias [10] (arreglo de vectores)

Estados finales: Teniendo en cuenta la poda, un estado final debe cumplir con la condición de que la cantidad de personas no supere su cupo de días.

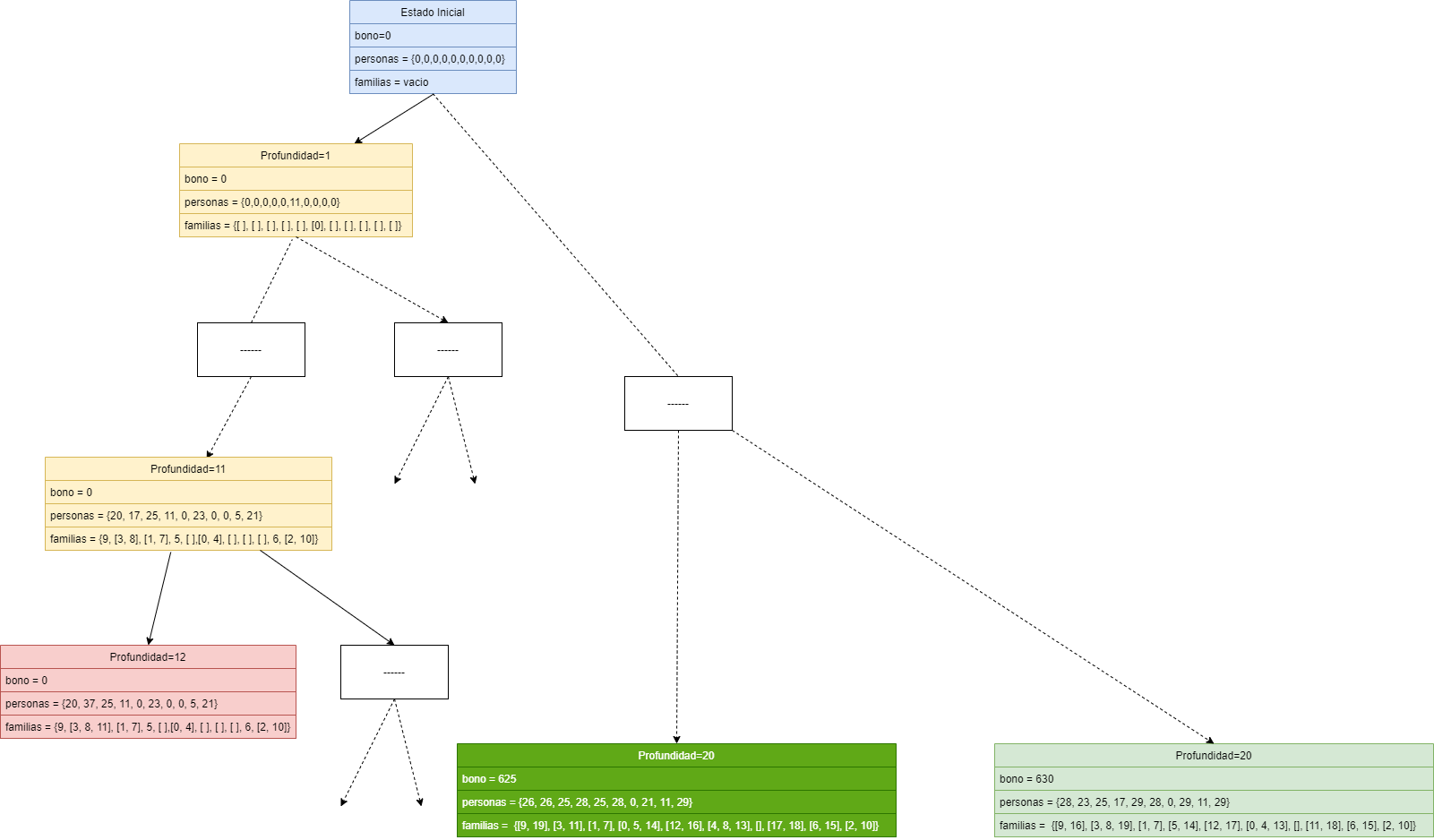
Estados solución: Deben ser estados finales y además la suma del bono debe ser mínima. Por ejemplo, en el primer dataset todo estado cuyo bono (bono\_obtenido) sea igual a 625 (resultado mínimo) va a ser solución.

Restricciones en la generación de estados: Que no se supere el cupo de personas por día. Cada familia es asignada a un único día para realizar la visita. Todas las familias son asignadas para realizar la visita en algunos de sus días preferidos.

**Seguimientos:**

*Seguimiento de estados del dataset 1:*

En el árbol incluí un estado que no supera la restricción (color rojo) solo a modo ilustrativo.



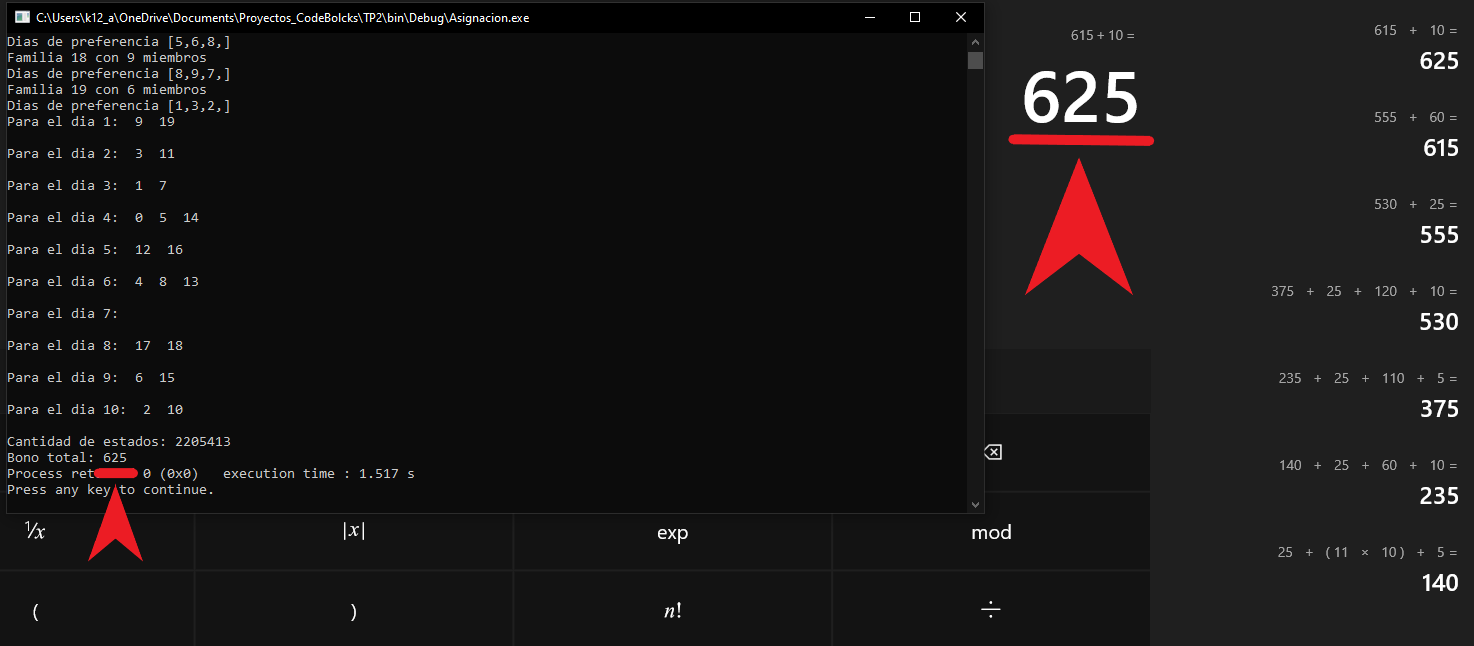
Estado inicial: Color azul.

Estado intermedio: Color amarillo.

Estado rechazado: Color rojo.

Estado solución: Color verde fuerte.

Estado final: Color verde suave.

*Seguimiento del bono del dataset 1, hecho en calculadora con historial de las cuentas.*

**Profundidad y anchura:**

La profundidad es igual a la cantidad de familias que tenga el dataset, por cada una se agrega un nodo en profundo.

La anchura por otro lado va a ser igual a n! con n = cantidad de familias. Esto es sin tener en cuenta la poda. Con poda depende de las características de la familia, dado a que si poseen muchos integrantes o no afecta al resultado.

**Comparación:**

Dataset-1:

* Con poda el tiempo de ejecución fue en promedio 0.295s, y tuvo una cantidad de estados = 8669.
* Sin poda el tiempo de ejecución fue en promedio de 0.295s, y tuvo una cantidad de estados = 2205413.

Dataset-2:

* Con poda el tiempo de ejecución fue en promedio 18.623s, y tuvo una cantidad de estados = 10279664.
* Sin poda no se pudo realizar.

**Conclusión:**

En este trabajo practico se realizó un algoritmo de backtracking y su solución la pensé basándome en el ejercicio N-reinas propuesto por la catedra de Análisis y Diseños de Algoritmos 2. Se puede observar una clase solución, de esta forma me pareció más sencilla la implementación dado que puedo agregarle métodos y encapsular objetivos individuales. También se implementó una clase parque en la que se encuentran las funciones para realizar el backtracking. Y de esta forma con invocar esta función en el programa principal se haya la solución. No me pareció necesario hacer la poda en una función aparte dadas las restricciones.

No se menciona el dataset 2 porque no pude realizar ninguna prueba sobre el mismo, mi equipo no soporta la ejecución de tantos estados, la única ejecución posible para el dataset 2 era quitándole familias hasta que solo quedaran 24, y la ejecución de este tardo 37 minutos. Esto se debe a que la cantidad de personas por familia es baja y esto provoca mayor compatibilidad de combinatoria (entran mas en un espacio de 30).