## **Optimización de Recorridos en Parques de Diversiones**

### (Combinación de Algoritmos Voraces + Backtracking para una Experiencia Eficiente)

### ****📌 Descripción del Proyecto****

Este sistema busca optimizar la experiencia de los visitantes en un parque de diversiones al planificar su recorrido de manera inteligente. Al ingresar sus preferencias y el tiempo disponible, el algoritmo recomendará las mejores atracciones a visitar minimizando los tiempos de espera y maximizando la diversión.

La solución combina **algoritmos voraces** y **backtracking**, permitiendo ajustar dinámicamente la ruta en caso de cambios en los tiempos de espera o disponibilidad de las atracciones.

### ****🔍 Técnicas Aplicadas****

1. **✅ Algoritmos Voraces:**
   * En cada paso, el sistema selecciona la atracción con la mejor relación **espera/disfrute**, priorizando aquellas que maximizan la experiencia del visitante dentro del tiempo disponible.
   * Se evalúan factores como tiempo de espera, duración de la atracción y distancia entre atracciones.
2. **🔄 Backtracking:**
   * Si el tiempo estimado de espera cambia drásticamente o si la selección actual no permite aprovechar al máximo el recorrido, el sistema reevaluará las opciones y ajustará el itinerario.
   * Esto garantiza una mayor flexibilidad y adaptabilidad en tiempo real.

### ****🎢 Ejemplo de Aplicación****

Imagina que un visitante tiene **4 horas** para disfrutar del parque y sus preferencias son **montañas rusas y juegos de agua**.

🔹 El sistema recopila datos en tiempo real sobre los tiempos de espera en cada atracción.  
🔹 Se genera una ruta optimizada que maximiza la cantidad de atracciones visitadas sin perder demasiado tiempo en filas.  
🔹 Si en algún momento una atracción sufre un aumento repentino en la espera, el sistema ajusta la ruta recomendada.

El usuario puede ver su recorrido actualizado en un **dashboard interactivo**, donde podrá modificar sus preferencias en cualquier momento.

### ****📊 Visualización y Herramientas Tecnológicas****

1. **📈 MATLAB**
   * Se generará un gráfico que muestra la relación entre **el tiempo de espera y la cantidad de atracciones visitadas**.
   * Permite visualizar cómo la estrategia optimiza la experiencia del visitante.
2. **📊 Power BI**
   * Se desarrollará un **dashboard interactivo** que muestre:  
     🔹 Estado de las atracciones en tiempo real.  
     🔹 Tiempo restante del recorrido.  
     🔹 Opciones alternativas para el usuario en caso de cambios.
   * El usuario podrá ajustar su ruta según sus preferencias o condiciones inesperadas.

### ****🚀 Impacto y Beneficios****

✅ **Mejor experiencia para los visitantes**, optimizando el tiempo en el parque.  
✅ **Menos tiempo en filas, más tiempo de diversión**.  
✅ **Sistema dinámico y adaptable** a los cambios en tiempo real.  
✅ **Visualización intuitiva** de los datos para mejorar la toma de decisiones.

**¿Qué cambia con 20 atracciones?**

✅ **Menos combinaciones a evaluar**, lo que hace que el algoritmo de Backtracking no sea demasiado lento.  
✅ **Fácil de simular datos de espera**, sin necesidad de integración con fuentes en vivo.  
✅ **Visualización más clara**, sin necesidad de mapas avanzados ni herramientas complejas.  
✅ **Factible de programar en MATLAB, Power BI o incluso una interfaz web simple**.

**🚀 Plan de Desarrollo (2 semanas)**

📌 **Semana 1: Algoritmo y Datos Simulados**  
✅ Definir un conjunto de 20 atracciones con tiempos de espera simulados.  
✅ Implementar un **algoritmo voraz** que seleccione la mejor atracción en cada paso.  
✅ Agregar **backtracking** para reajustar la ruta si el tiempo de espera cambia.  
✅ Hacer pruebas en **consola** para verificar que la ruta generada es eficiente.

📌 **Semana 2: Visualización y Optimización**  
✅ Crear una **visualización de la ruta** (gráfico en MATLAB o dashboard en Power BI).  
✅ Agregar la opción de **ajustar preferencias** (por ejemplo, priorizar montañas rusas o juegos de agua).  
✅ Hacer **pruebas finales** y mejorar la eficiencia del código.

**🛠️ Herramientas que Puedes Usar**

1. **Lenguaje de Programación:** Python (más flexible) o MATLAB (si solo te enfocas en visualización).
2. **Estructura de Datos:** Listas o Grafos para representar atracciones y conexiones.
3. **Visualización:** Power BI para dashboards interactivos o MATLAB para gráficos simples.