ESTRUCTURA DE DATOS

PROYECTOS 2025-2

Docente: Dr. Eric Gustavo Coronel Castillo

PROPÓSITO

Desarrollar soluciones a problemas reales aplicando listas enlazadas (simples, dobles, circulares), pilas, colas, árboles binarios (ABB) y grafos (BFS, DFS, ordenamiento topológico, caminos mínimos y MST), con capacidad para explicar el porqué de cada estructura y algoritmo.

ALCANCE TÉCNICO

Implementación en C# (.NET 6+ o superior), sin utilizar colecciones de alto nivel para sustituir las estructuras requeridas (Por ejemplo, no usar List<T> o LinkedList<T> como almacenamiento principal), salvo para utilidades externas (Por ejemplo, carga de datos). Las estructuras evaluadas deben ser de autoría propia.

ENTREGABLES

1. Código fuente en C#, con estructuras implementadas por el equipo (no sustituir por colecciones de alto nivel).
2. Informe técnico: problema, diseño y justificación de EDD, decisiones clave, análisis de desempeño (empírico y/o Big-O), pruebas y conclusiones.
3. Video explicativo (mínimo 10 min): problema, diagrama de EDD, decisiones, demostración de ejecución, resultados y conclusiones. Con cámara activa y participación de todos los integrantes.
4. Demostración en vivo (máximo 10 min): ejecución guiada con dataset representativo, casos borde y defensa técnica.
5. Guía de ejecución, datasets y material de presentación (diapositivas/guion).

GLOSARIO DE TÉRMINOS

Este glosario resume los términos usados en los proyectos y la rúbrica. Lenguaje claro, definiciones breves y un ejemplo de uso en el curso.

TÉRMINO	DEFINICIÓN	EJEMPLO EN EL CURSO
EDD	Estructuras de Datos. Conjunto de modelos y técnicas para organizar y manipular información (listas, pilas, colas, árboles, grafos).	Curso completo.
Nodo	Unidad básica de una lista: contiene datos y enlaces (referencias) a otros nodos.	Listas enlazadas.
Cabeza (head) / Cola (tail)	Primer y último nodo de una lista o cola, respectivamente.	Listas/colas.
Lista simplemente enlazada	Cada nodo apunta al siguiente. Recorrido hacia adelante.	Implementación de colas, historiales.
Lista doblemente enlazada	Cada nodo apunta al siguiente y al anterior. Navegación en ambos sentidos.	Listas de reproducción, pedidos en salón.
Lista circular	El último nodo enlaza con el primero. Útil para rotaciones.	Round-robin de agentes/repartidores.
Pila (stack)	LIFO: el último en entrar es el primero en salir.	Bandejas en aeropuerto; garaje en una sola vía.
Cola (queue)	FIFO: el primero en entrar es el primero en salir.	Filas de atención, tickets por canal.
Cola prioritaria	Atiende según prioridad. En empates se puede respetar el orden de llegada (estabilidad).	Triaje hospitalario.
Cola de apoyo (overflow)	Cola extra que se habilita cuando la cola principal supera un umbral (longitud o espera).	Atención municipal.
Envejecimiento (aging)	Regla que garantiza turnos a menor prioridad tras X atenciones preferentes o T minutos de espera.	Banco con preferencia a adultos mayores.
BFS (Búsqueda en Anchura)	Recorre por niveles y obtiene distancias mínimas en número de tramos en grafos no ponderados.	Rutas peatonales en campus.

DFS (Búsqueda en Profundidad)	Explora lo más profundo antes de retroceder. Útil para detección de componentes y ciclos.	Análisis de grafos.
Ordenamiento topológico	Orden de tareas/cursos en un DAG sin violar prerequisitos (algoritmo de Kahn o DFS).	Plan de estudios.
DAG	Grafo dirigido acíclico.	Prerequisitos de cursos.
Dijkstra	Caminos mínimos con pesos no negativos.	Logística urbana (rutas de reparto).
MST (Árbol de Expansión Mínima)	Subconjunto de aristas que conecta todos los nodos con costo total mínimo.	Red de sedes (Kruskal/Prim).
Union-Find (DSU)	Estructura para conjuntos disjuntos: une y consulta componentes rápidamente.	Kruskal para MST.
Grafo no ponderado	Todas las aristas tienen el mismo costo.	Rutas por número de tramos (BFS).
Grafo ponderado	Las aristas tienen costos/pesos distintos (distancia, tiempo).	Rutas con tiempos reales (Dijkstra).
Grafo disperso	Pocas aristas respecto del máximo posible; conviene lista de adyacencia.	Mapas de calles/campus.
Lista de adyacencia	Representación de un grafo con las vecindades de cada nodo.	BFS/DFS/Dijkstra.
Predecesor	Nodo desde el que se descubrió otro; permite reconstruir rutas.	Rutas mínimas.
Estabilidad	Mantener el orden de llegada ante empates de prioridad.	Triaje o colas por nivel.
Timestamp	Marca de tiempo; se usa para desempates o métricas.	Llegadas y tiempos de servicio.
SLA	Objetivo de servicio (p. ej., % de atenciones ≤ 10 min).	Panel de métricas.
p90 / p95 / p99	Percentiles: valor que el 90/95/99% de los casos no supera.	Reportes de tiempos de espera.
Mediana (p50)	La mitad de los casos está por debajo y la otra mitad por encima.	Resumen robusto de tiempos.
Tiempo medio (promedio)	Suma de tiempos dividido entre el número de casos.	Indicador general de desempeño.

Big-O / Notación asintótica	Forma de acotar el crecimiento del costo (O, Ω, Θ).	Análisis de operaciones (opcional).
O(1) / O(n)	Constante / lineal: costo no depende de n / crece con n.	Insertar cola vs. recorrer lista.
CI local	Pipeline de integración continua ejecutado en la máquina del equipo (build + pruebas + métricas).	Reproducibilidad de entregas.
.NET Stopwatch	Cronómetro de alta resolución para medir tiempos.	Mediciones empíricas.
BenchmarkDotNet	Framework de micro-benchmarks para .NET.	Mediciones precisas (opcional).
Dataset	Conjunto de datos para pruebas (CSV/JSON/Excel).	Simulaciones y reportes.
CSV	Archivo de valores separados por comas, usualmente con encabezado.	Datasets de llegadas/servicios.
Round-robin	Asignación cíclica y equitativa de turnos.	Agentes de call center; repartidores.
Umbral	Límite a partir del cual se activa una política.	Activar cola de apoyo.
Caso borde	Situación extrema o poco frecuente que debe manejarse correctamente.	Cola vacía; stock insuficiente.

PROYECTO 1
Gestión de colas en un banco con preferencia para adultos mayores

Objetivo aplicado

Diseñar la atención justa en cajas con prioridad para clientes de mayores de 60 años sin generar desatención para el resto.

Contexto

Un banco atiende clientes con categorías: Preferente (mayores de 60 años), Embarazadas/Discapacidad, y General. Hay varias cajas y reglas de prioridad.

Estructuras y algoritmos a usar

- Colas basadas en lista simplemente enlazada por categoría (Preferente, Prioritaria, General).
- Para evitar esperas prolongadas, tras X atenciones preferentes consecutivas se atenderá al primer cliente de la cola General (si lo hubiera).
- Estadísticas de espera por categoría.

Requisitos funcionales mínimos

- Llegada de clientes (DNI, edad, categoría, hora llegada).
- Asignación a caja según prioridad y disponibilidad.
- Regla de equidad: por ejemplo, máximo 2 preferentes seguidos antes de atender 1 general si hay en espera.
- Reporte: tiempo medio de espera por categoría, utilización de cajas.

Cómo explicar la aplicación de EDD

- Justificar por qué múltiples colas por categoría facilitan reglas de prioridad.
- Demostrar cómo la lista enlazada permite inserción en cola en tiempo casi constante.
- **Explica el envejecimiento:** qué mecanismo usan para llevar la cuenta y cómo asegura que la cola General también avance (sin dejar a nadie

esperando demasiado).

Datos/insumos sugeridos

- CSV con 1 000–10 000 llegadas (edad, hora, categoría).
- Configuración: número de cajas y política de equidad (por ejemplo, 2P:1G).

Pruebas mínimas

- Escenario saturado con muchos preferentes: verificar equidad y tiempos.
- Escenario sin preferentes: operación normal.
- Caja fuera de servicio temporal: redistribución correcta.

Extensiones opcionales

- Asignación dinámica de cajas según demanda por franja.
- Análisis de sensibilidad (variar reglas y comparar métricas).

PROYECTO 2
Triaje de emergencia hospitalaria (urgencias con prioridad clínica)

Objetivo aplicado

Organizar la atención según niveles de gravedad (T1–T5) y tiempo máximo de espera.

Contexto

El servicio de emergencia clasifica pacientes por triage; los más graves deben ser atendidos antes sin descuidar casos leves.

Estructuras y algoritmos a usar

- Cola que atiende primero a los más graves; si dos pacientes tienen el mismo nivel, pasa primero quien llegó antes (por orden de llegada).
- Lista de espera secundaria para derivaciones o reevaluación.
- Contadores de tiempo de espera por nivel.

Requisitos funcionales mínimos

- Ingreso de paciente (ID, triage 1–5, hora).
- Atención priorizando menor triage; en empates, más antiguo (orden de llegada).
- Alerta si tiempo de espera excede umbral por nivel.

Cómo explicar la aplicación de EDD

- Justificar inserción ordenada estable para la cola prioritaria.
- Explicar por qué no se usa un arreglo con reordenamientos costosos.
- Describir cómo se recalcula prioridad cuando cambia el triage.

Datos/insumos sugeridos

- Dataset de llegadas y tiempos de servicio simulados por nivel.

Pruebas mínimas

- Picos de llegada; reevaluación que cambia triage.
- Tiempo excedido produce alerta y reordenamiento.

Extensiones opcionales

- Múltiples médicos/boxes en paralelo (servidores M/M/c).

PROYECTO 3
Atención en municipalidad con múltiples servicios

Objetivo aplicado

Balancear filas por servicio (licencias, pagos, informes) manteniendo métricas de espera y SLA.

Contexto

La municipalidad tiene ventanillas por servicio y demanda variable a lo largo del día.

Estructuras y algoritmos a usar

- Múltiples colas (listas enlazadas) por servicio.
- **Cola de apoyo:** si la cola principal supera el umbral, se abre una cola extra. Los nuevos usuarios van a esa cola y se intercalan turnos (uno de la cola extra y uno de la principal) hasta normalizar; luego se cierra la cola extra.
- Estadísticas rodantes en una lista circular (ventanas de tiempo).

Requisitos funcionales mínimos

- Registro de llegada con tipo de trámite.
- Asignación a ventanilla más adecuada o a cola de apoyo si hay saturación.
- **Dashboard de métricas:** tiempo medio, mediana, máximo y p90 (percentil 90: el 90% no supera este valor)

Cómo explicar la aplicación de EDD

- Defender el uso de colas por servicio frente a una bolsa única.
- Explicar la lista circular para métricas en ventanas móviles.

Datos/insumos sugeridos

- CSV con llegadas por franja horaria.
- Capacidad por ventanilla configurable.

Pruebas mínimas

- Saturación en un servicio y balance por colas de apoyo.
- Cambios abruptos de demanda (evento).

Extensiones opcionales

- Prioridades por tipo de trámite con reglas de equidad.

PROYECTO 4
Almacén de perecibles: mezcla de LIFO y FIFO

Objetivo aplicado

Modelar depósitos con apilamiento (LIFO) y despacho por FIFO según producto.

Contexto

Un supermercado maneja productos con diferentes políticas: bebidas (LIFO por pallets), lácteos (FIFO por fecha).

Estructuras y algoritmos a usar

- Pila (stack) para pallets en cámaras (LIFO).
- Cola (queue) para lotes con fecha de caducidad (FIFO).
- Lista para catálogo y búsqueda básica.

Requisitos funcionales mínimos

- Ingreso de lotes con fecha; salida según política del producto.
- Alertas por caducidad próxima; inventario por categoría.

Cómo explicar la aplicación de EDD

- Justificar por qué algunos flujos son LIFO y otros FIFO.
- Explicar cómo se evitan pérdidas por caducidad (FIFO).

Datos/insumos sugeridos

- CSV de entradas y salidas por día; fechas y cantidades.

Pruebas mínimas

- Picos de entrada; pedidos simultáneos.
- Lotes vencidos generan alerta y bloqueo de salida.

Extensiones opcionales

- Reabastecimiento automático por mínimos.

PROYECTO 5
Estacionamiento en una sola vía con carril auxiliar

Objetivo aplicado

Gestionar entradas y salidas en un estacionamiento lineal usando una pila auxiliar para extraer autos intermedios sin bloquear el flujo.

Contexto

Un garaje en línea tiene 10–50 plazas. Para sacar un auto que no está al final, se deben mover temporalmente los autos posteriores.

Estructuras y algoritmos a usar

- Pila principal para los autos estacionados (LIFO).
- Pila auxiliar para mover y reinsertar autos cuando sale uno intermedio.
- Lista simplemente enlazada para el historial de movimientos (entradas y salidas).

Requisitos funcionales mínimos:

- Registrar entrada (placa, hora) y validar capacidad.
- Procesar salida por placa: mover autos a la pila auxiliar hasta encontrarlo, calcular tiempo y costo, y reponer los autos.
- Reportes: ocupación, tiempo promedio de estancia, movimientos por salida.

Cómo explicar la aplicación de EDD

- Justificar por qué el flujo físico del garaje se modela como pila (LIFO).
- Explicar el uso de la pila auxiliar para salidas intermedias.
- Comentar el impacto: salidas del fondo son directas; intermedias requieren más movimientos.

Datos/insumos sugeridos

- CSV de eventos: tipo (entrada/salida), placa, hora (HH:MM).
- Capacidad del garaje y tarifa por hora.

Pruebas mínimas

- Salida del último auto (caso simple).
- Salida del primero (requiere mover varios autos a la auxiliar).
- Garaje lleno frente a nueva entrada (rechazo).
- Salida de placa inexistente (manejo de error).

Extensiones opcionales

- Reservas de plazas (lista de autorizados).
- Tarifa diferenciada por franja horaria.

PROYECTO 6
Centro de llamadas multicanal (teléfono, chat y correo)

Objetivo aplicado

Asignar tickets de distintos canales a agentes de manera equilibrada, respetando orden de llegada y pausas del personal.

Contexto

El centro recibe consultas por teléfono, chat y correo. Debe distribuirlas entre agentes activos y manejar pausas/retornos.

Estructuras y algoritmos a usar

- Colas FIFO separadas por canal (teléfono, chat, correo).
- Lista circular de agentes activos para repartir turnos (round-robin).
- Lista doble para pausar y reinsertar agentes sin perder su posición relativa.

Requisitos funcionales mínimos

- Registrar llegada de tickets con canal, cliente y hora.
- Asignar automáticamente al siguiente agente disponible según round-robin.
- Pausar y reactivar agentes manteniendo la rotación.
- Métricas: tiempos de espera por canal y carga por agente.

Cómo explicar la aplicación de EDD

- Cómo las colas aseguran el orden de llegada por canal.
- Por qué la lista circular reparte el trabajo de forma justa.
- Uso de lista doble para extraer y devolver agentes a la rotación.

Datos/insumos sugeridos

- CSV de llegadas por canal.
- Lista de agentes con horarios de pausa simulados.

Pruebas mínimas

- Saturación de un canal (muchos chats) y asignación equilibrada.
- Agente en pausa y retorno; la rotación se mantiene.
- Llegadas simultáneas en varios canales.

Extensiones opcionales

- Preferencia por canal (por ejemplo, chat antes que correo).
- Envejecimiento para correos con mucha espera.

PROYECTO 7
Farmacia hospitalaria: preparación de recetas

Objetivo aplicado

Atender recetas con prioridad a urgentes y validar cada preparación con un checklist estructurado.

Contexto

La farmacia recibe recetas “Urgentes” y “Normales”. Se prioriza urgentes, se verifica stock y se documenta el proceso.

Estructuras y algoritmos a usar

- Dos colas FIFO: Urgentes y Normales (atender primero Urgentes).
- Pila de checklist por receta (cada paso se “pop” al completarse).
- Lista simplemente enlazada para inventario básico (medicamento, cantidad).

Requisitos funcionales mínimos

- Registrar receta (ID, tipo, hora) y enviarla a la cola correspondiente.
- Preparar receta: generar y cumplir el checklist; controlar stock.
- Si falta stock, mover la receta a “pendientes por reposición”.
- Reportes: tiempos de espera por tipo, completadas y pendientes.

Cómo explicar la aplicación de EDD

- Por qué dos colas simplifican la prioridad.
- Cómo la pila asegura no saltar pasos del checklist.
- Cómo la lista enlazada facilita actualizar inventario.

Datos/insumos sugeridos

- CSV de recetas (ID, tipo, hora).
- CSV de inventario (medicamento, cantidad).

Pruebas mínimas

- Oleada de urgentes seguida de normales.
- Paso fallido por falta de stock.
- Checklist completo para varias recetas.

Extensiones opcionales

- Reposición automática por mínimos.
- Lotes y fechas de caducidad (lista ordenada por fecha).

PROYECTO 8
Restaurante: pase de cocina y reparto

Objetivo aplicado

Coordinar pedidos desde cocina hasta entrega, optimizando el retiro de platos y la asignación de repartidores.

Contexto

Un restaurante gestiona pedidos de salón y delivery. La cocina prepara y deposita platos en un pase de cocina.

Estructuras y algoritmos a usar

- Cola de pedidos para cocina (FIFO).
- Pila de “platos listos” en el pase (retiro inmediato del último listo).
- Lista circular de repartidores/mozos para turnos de entrega.
- Lista doble para navegar pedidos de salón por mesa y estado.

Requisitos funcionales mínimos

- Registrar pedido y pasarlo a cocina.
- Mover a “plato listo” al terminar y retirarlo desde la pila.
- Asignar entregas según lista circular.
- Reportes: tiempo en cocina, en pase y entregas por persona.

Cómo explicar la aplicación de EDD

- La cola conserva el orden de llegada a cocina.
- La pila hace rápido el retiro de platos listos.
- La lista circular reparte turnos de entrega de forma simple.

Datos/insumos sugeridos

- CSV de pedidos (hora, tipo, mesa/dirección).

Pruebas mínimas

- Alta demanda: muchos platos listos en poco tiempo.
- Rotación de repartidores con pausas.
- Pedidos de salón y delivery simultáneos.

Extensiones opcionales

- Prioridad para pedidos express con segunda cola.
- Panel en tiempo real con estados.

PROYECTO 9
Control de seguridad en aeropuerto: bandejas y filas

Objetivo aplicado

Agilizar el control gestionando filas de pasajeros y el uso de bandejas de forma ordenada.

Contexto

En seguridad, los pasajeros forman filas y usan bandejas para sus pertenencias; las bandejas se apilan y se reutilizan.

Estructuras y algoritmos a usar

- Cola de pasajeros (FIFO) por línea de control.
- Pila de bandejas por línea (LIFO) para entrega y devolución rápida.
- Lista simplemente enlazada para registrar incidencias (objetos prohibidos, revisiones).

Requisitos funcionales mínimos

- Registrar llegada de pasajeros y asignarlos a una línea.
- Entregar bandejas desde la pila y devolverlas al finalizar.
- Marcar incidencias y generar reportes.
- Métricas: espera por línea, uso de bandejas, incidencias por hora.

Cómo explicar la aplicación de EDD

- Por qué las filas de pasajeros se modelan como colas.
- Por qué las bandejas se gestionan como pilas.
- Cómo la lista enlazada facilita el registro de incidencias.

Datos/insumos sugeridos

- CSV de llegadas por hora y líneas disponibles.

Pruebas mínimas

- Pico de llegadas con bandejas limitadas.
- Desbalance entre líneas (una más rápida que otra).
- Múltiples incidencias y su registro.

Extensiones opcionales

- Balanceo: mover pasajeros a otra línea si se supera un umbral.
- Estadísticas por tipo de incidencia.

PROYECTO 10
Rutas peatonales más cortas en el campus (no ponderado)

Objetivo aplicado

Encontrar caminos mínimos en pasos (aristas unitarias).

Contexto

Un mapa peatonal representado como grafo no ponderado.

Estructuras y algoritmos a usar

- Grafo con listas de adyacencia (nodos = intersecciones).
- BFS para distancias y reconstrucción de ruta.

Requisitos funcionales mínimos

- Cargar grafo; consultar ruta entre pares.
- Detectar zonas inaccesibles y componentes.

Cómo explicar la aplicación de EDD

- Justificar listas de adyacencia en grafos dispersos.
- Explicar por qué BFS sirve en aristas de costo uniforme.

Datos/insumos sugeridos

- Archivo con vértices y aristas.

Pruebas mínimas

- Puentes cerrados; múltiples rutas equivalentes.

Extensiones opcionales

- Visualización simple de la ruta.

PROYECTO 11
Plan de estudios con prerrequisitos (ordenamiento topológico)

Objetivo aplicado

Calcular un orden de cursado y detectar ciclos.

Contexto

Estructurar los ciclos de un programa académico respetando prerequisitos.

Estructuras y algoritmos a usar

- DAG con listas de adyacencia.
- Ordenamiento topológico (Kahn o DFS).

Requisitos funcionales mínimos

- Cargar cursos y prerequisitos.
- Sugerir niveles/periodos de cursado.

Cómo explicar la aplicación de EDD

- Explicar aciclicidad y manejo de ciclos detectados.
- Justificar elección de Kahn o DFS.

Datos/insumos sugeridos

- CSV de cursos (30–50) y prerequisitos.

Pruebas mínimas

- Caso con ciclo; múltiples órdenes válidos.

Extensiones opcionales

- Asignación de créditos por periodo.

PROYECTO 12
Red de sedes: Árbol de Expansión Mínima (MST)

Objetivo aplicado

Minimizar el costo de conectar todas las sedes.

Contexto

Una empresa quiere desplegar enlaces entre sedes físicas con costo mínimo.

Estructuras y algoritmos a usar

- Grafo ponderado no dirigido.
- Kruskal (Union-Find) o Prim con estructura simple de prioridad.

Requisitos funcionales mínimos

- Construir grafo desde archivo; validar conectividad.
- Listar aristas del MST y su costo total.

Cómo explicar la aplicación de EDD

- Justificar la elección del algoritmo (densidad del grafo, facilidad de implementación).
- Explicar por qué el MST no necesariamente optimiza latencias entre pares.

Datos/insumos sugeridos

- Instancias pequeñas y medianas; coordenadas opcionales.

Pruebas mínimas

- Grafos no conectados deben reportarse.
- Pesos iguales → múltiples MST posibles.

Extensiones opcionales

- Visualización del MST.

PROYECTO 13
Biblioteca universitaria: índice por ABB

Objetivo aplicado

Búsqueda eficiente por ISBN/título con recorridos y estadísticas.

Contexto

Construir un índice de libros que permita insertar, buscar, eliminar y listar ordenado.

Estructuras y algoritmos a usar

- Árbol binario de búsqueda (ABB).
- Recorridos in-order y por niveles.

Requisitos funcionales mínimos

- Cargar libros; evitar duplicados.
- Eliminar manejando hoja/1 hijo/2 hijos.

Cómo explicar la aplicación de EDD

- Explicar cómo se mantiene el invariante del ABB.
- Discutir efecto del orden de inserción en la altura.

Datos/insumos sugeridos

- CSV de libros (1k–10k registros).

Pruebas mínimas

- Inserciones ordenadas vs aleatorias; contraste de alturas.

Extensiones opcionales

- Exportación a DOT y visualización.

PROYECTO 14
Logística urbana: rutas de reparto de menor tiempo

Objetivo aplicado

Calcular rutas de costo mínimo (tiempo/distancia) en un grafo dirigido.

Contexto

Simular entregas en una ciudad con restricciones horarias.

Estructuras y algoritmos a usar

- Grafo dirigido/ponderado con listas de adyacencia.
- Dijkstra (con cola de prioridad implementada por ustedes).

Requisitos funcionales mínimos

- Carga de grafo y consulta de rutas.
- Reconstrucción de caminos y reporte del tiempo total.

Cómo explicar la aplicación de EDD

- Justificar Dijkstra para pesos no negativos.
- Explicar la estructura de prioridad usada y su impacto práctico.

Datos/insumos sugeridos

- Instancias 50–500 nodos; demandas de entrega.

Pruebas mínimas

- Calles bloqueadas/desconectadas; múltiples rutas equivalentes.

Extensiones opcionales

- Ventanas horarias o restricciones por zonas.