선형자료구조
스택, 큐

NEXTERS

Contiguous vs. Linked

- 자료 구조는 포인터나 배열을 통해 연결되어 있거나 인접 할 수 있다.
- 인접한 자료구조 : 배열, 행렬, 힙, 해시 테이블
- 연결된 자료구조: 리스트, 트리, 그래프

배열

- 상수시간 접근 인덱스 : 특정한 메모리 주소가 주어지기 때문에 인덱스를 통해 상수시간 안에 접근 할 수 있다.
- 공간 효율성 : 구현에 추가의 포인터 없이 순수 데이터만 필요하다.
- 메모리 지역성 : 모든 요소가 메모리에 지역적으로 분포하기 때문에 컴퓨터 아키텍처상 빠른 스피드 캐시가 이용이 가능하다.

동적 배열

- 배열을 이용하여 자료구조 크기가 커져도 동적으로 사이즈가 커질 수 있도록 고안된 배열
- 배열의 크기를 변경하는데 크기 N에 비례하는 시간이 걸림
- 주어진 원소를 추가하는데 맨 마지막에 원소를 추가함으로 상수 시간이 걸림

동적 배열의 재할당 전략

- 크기를 늘려야 할 때마다 두 배의 크기로 재 할당
- 재 할당에는 이전 요소를 복사하는 행위도 포함된다.
- 시간 복잡도

$$\sum_{i=1}^{\lg n} i \cdot n/2^i = n \sum_{i=1}^{\lg n} i/2^i \leq n \sum_{i=1}^{\infty} \frac{i}{2^i} = 2n$$

연결 리스트

- 삽입, 삭제가 배열보다 간단하다
- 큰 자료의 경우 포인터를 이동하는 것이 아이템 전체를 이동 하는 것 보다 빠르다.
- 구현을 위해 포인터 공간이 추가로 필요하다
- 랜덤 접근에 효율적이지 않다.

큐

- 한쪽에서 자료를 넣고 반대 쪽에서 자료를 꺼낼 수 있다
- 선입 선출 (FIFO : First In First Out)
- Enqueue: 아이템을 넣는 과정
- Dequeue : front 아이템을 가져오는 과정 (가져온 후 제거)
- 데크 : 양쪽 끝에서 자료를 넣고 뺄 수 있는 구조

스택

- 한쪽 끝에서만 자료를 넣고 뺄 수 있다.
- 후입 선출 (LIFO , Last In First Out)
- Push : 자료를 넣는 과정
- Pop : 자료를 가져오는 과정

큐, 스텍, 데크의 구현

- 리스트를 이용한 구현
  - 양쪽 끝에서 추가 와 삭제가 상수시간에 가능하기 때문에 쉽게 구현가능
  - 노드의 할당과 삭제 포인터 따라가는 시간 때문에 최적화 방법은 아님
- 동적 배열을 이용한 구현
  - 스택은 쉽게 구현이 가능하지만 큐와 데크의 경우 앞 부분의 자료 처리가 문제 가 됨
  - 환형 구조를 사용해 낭비되는 부분을 해결

조세푸스 문제 (JOSEPHUS)

- N 명의 사람들이 원형으로 서있고 K번 째의 사람들이 순서대로 죽을 때 마지막에 살아 남는 두 사람의 자리를 구하는 문제

- 원형으로 연결된 연결 리스트를 통해 구현 한다.

✓ 문제

https://algospot.com/judge/problem/read/JOSEPHUS

✓ 풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter18/Josephus.cpp

https://github.com/Nexters/algorithmStudy/blob/master/yongseongkim/chapter_18_19/josephus.py

https://github.com/Nexters/algorithmStudy/blob/master/Hsue/JOSEPHUS/josep.py

짝이 맞지 않는 괄호 문제(BRACKETS2)

- 각 괄호의 쌍들의 문자열이 있을 때 잘 맞는지 확인 하는 문제
- 스택을 이용해 문자열을 제거해 나가면서 쌍이 맞는지 확인한다.

✓ 문제
https://algospot.com/judge/problem/read/BRACKETS2

✓ 풀이코드
https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter19/Brackets2.cpp
https://github.com/Nexters/algorithmStudy/blob/master/yongseongkim/chapter_18_19/brackets2.py
https://github.com/Nexters/algorithmStudy/blob/master/Hsue/BRACKETS2/bracket.cpp

외계 신호 분석 (ITES)

- 1 <= N <= 50,000,000 의 크기의 부분 수열 중 특정 합이 되는 부분 수열을 찾는다.

- 입력을 한번에 담을 수 없기 때문에 부분적으로 큐에 담으면서 부분 수열을 구한다

✓ 문제

https://algospot.com/judge/problem/read/ITES

✓ 풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter19/Ites.cpp

https://github.com/Nexters/algorithmStudy/blob/master/yongseongkim/chapter_18_19/ites.py

https://github.com/Nexters/algorithmStudy/blob/master/Hsue/ITES/ites.cpp