자료구조 (Data Structure)
1주차: 배열 리스트

자료구조란?

- 프로그램에 데이터를 저장하는 효율적 방법
- 초기 프로그래머들은 자료구조를 직접 개발
- 보편적 자료구조들을 모아서 교과서가 편찬됨

강의 소개

- 교재: Fundamentals of Data Structures in C
  by Horowitz, Sahni & Anderson-Freed
- 참고웹: https://moca.ajou.ac.kr/ 자료구조 모듈
- 방식: 매주 하나의 주제를 다룬다
  화요일 - 동작 방식 개요
  금요일 - C 코드로 구현

강의 계획

- 9월: 일차원 자료구조
- 10월: 계층적 자료구조
- 11월: 자유도가 높은 자료구조
- 12월: 탐색에 효율적인 자료구조

9월 강의 계획

- 1주차: 모든 데이터에 접근 가능 (배열 리스트)
- 2주차: 모든 데이터에 접근 가능 (링크드 리스트)
- 3주차: 마지막 데이터만 접근 가능 (스택)
- 4주차: 첫 데이터만 접근 가능 (큐)

10월 강의 계획

- 5주차: 뿌리에서 뻗어나가듯 데이터 연결 (트리)
- 개천절, 추석 연휴 (10/3, 7)
- 6주차: 뿌리에서 멀수록 값이 작음/큼 (힙)
- 중간고사
- 7주차: 저장된 데이터를 정렬하기

11월 강의 계획

- 8주차: 데이터 간 연결에 제약 없음 (그래프)
- 9주차: 그래프의 최단 경로
- 10주차: 그래프를 트리로 변환
- 11주차: 배열 접근 방법 확장 (해시 테이블)

12월 강의 계획

- 12주차: 왼쪽은 작고 오른쪽은 큰 이진탐색트리
- 13주차: 항상 균형을 이루는 레드블랙트리
- 기말고사

성적 평가 방식

- 출석: 15% (미니 퀴즈 제출 여부)
  미니 퀴즈 점수는 동점자 성적에 반영
- 과제: 15% (C 언어 프로그래밍)
- 중간/기말고사: 35% (PPT에서 출제)

리스트

- 정의: 데이터들에 1번부터 순서를 붙여서 저장한다
- 기능:
  - INSERT(k, x): k번째 위치에 원소 x 추가
  - DELETE(k): k번째 원소 삭제
  - RETRIEVE(k): k번째 원소 반환

배열 리스트

- 데이터를 배열에 순서대로 저장한다
- 저장된 데이터 수와 최대 저장량을 기록해둔다
- data: 배열
- size: 저장된 데이터 수
- capacity: 최대 저장량

배열 인덱스 규칙

- 편의상 배열 인덱스는 1부터 시작한다
- 따라서 배열 크기는 capacity + 1 로 만든다
- 인덱스 0은 사용하지 않고 비워둔다

INITIALIZE(A, cap)

- cap개 데이터를 저장가능하게 A를 초기화한다
- 조건: cap > 0
- 동작: A.data ← 새로운 배열(크기 : cap + 1)

  A.size ← 0

  A.capacity ← cap

INSERT(A, k, val)

- A의 k번째 위치($1 \le k \le A.size + 1$)에 val을 추가한다
- 조건: $A.size < A.capacity$
- 동작: A.data[A.size부터 k까지 역순]를 뒤로 이동

  $A.data[k] \leftarrow x$

  $A.size \leftarrow A.size + 1$

INSERT(A, k, val)의 시간 효율

- k == 0일 때:

  배열에 대입 A.size + 1번

- k == A.size + 1일 때:

  배열에 대입 1번

- 평균:

$$\frac{1 + \cdots + (n+1)}{n} = \frac{(n+1)(n+2)}{2n} \sim n$$

DELETE(A, k)

- A의 k번째 위치($1 \le k \le A.size$)의 원소를 삭제한다
- 조건: $A.size > 0$
- 동작: A.data[k+1부터 A.size까지]를 앞으로 이동

  $A.size \leftarrow A.size - 1$

DELETE(A, k)의 시간 효율

- k == 0일 때:

  배열에 대입 A.size - 1번

- k == A.size일 때:

  배열에 대입 0번

- 평균:

$$\frac{0 + \cdots + (n-1)}{n} = \frac{n(n-1)}{2n} \sim n$$

RETRIEVE(A, k)

- A의 k번째 위치($1 \le k \le A.size$)의 원소를 반환한다
- 조건: $A.size > 0$
- 동작: A.data[k]를 반환

RETRIEVE(A, k)의 시간 효율

- 데이터 주소 계산, 저장된 값 읽기 (상수 시간)

가변 크기 배열 리스트

- INSERT를 할 때:

  배열이 가득 찼으면,

  더 큰 배열을 새로 만들고,

  새 배열에 기존 데이터를 복사한다

  원래 INSERT 작업을 한다

RESIZE(A)

- A의 capacity를 두배 늘리고, 배열을 새로 만든다
- 기존 데이터는 새로운 배열에 모두 복사한다
- 동작: A.capacity ← A.capacity * 2

  tmp ← 새로운 배열(크기 : A.capacity + 1)

  A.data에서 tmp로 A.size개 데이터 복사

  A.data ← tmp

INSERT_VAR(A, k, x)

- A의 k번째 위치(1 ≤ k ≤ A.size + 1)에 x를 추가한다

- 동작: 만약 A.size == A.capacity면 RESIZE(A)

  INSERT(A, k, x)

n회 INSERT할 동안 RESIZE에 쓴 시간

- 가정: 배열 용량이 가득 차면 r배로 늘린다 (r > 1)

  n번째 추가할 때 RESIZE가 실행되었다

  n은 아주 크다

- 복사된 횟수:

$$n + \frac{n}{r} + \cdots = \frac{rn}{r-1} \sim n$$

RESIZE 배율에 따른 시간, 공간 효율

- 3배: 복사횟수 ≈ 1.5n, 메모리 ≈ 3n, 곱하면 $4.5n^2$
- 2배: 복사횟수 ≈ 2n, 메모리 ≈ 2n, 곱하면 $4n^2$
- 1.5배: 복사횟수 ≈ 3n, 메모리 ≈ 1.5n, 곱하면 $4.5n^2$
- → 2배씩 증가하는 것이 시간 × 공간 비용 최소

예제: 다항식 저장하기

- $2x^2 + 3x$를 저장하려면,
- 1. 항 리스트를 만든다 (계수와 지수 저장)
- 2. 항 리스트를 크기 2로 초기화한다
- 3. 항 리스트에 계수 2, 지수 2을 추가한다
- 4. 항 리스트에 계수 3, 지수 1을 추가한다

고정 크기 항 리스트

- coeff: 계수를 저장할 배열
- exp: 지수를 저장할 배열
- size: 저장된 항의 개수
- capacity: 최대 저장량

INITIALIZE(A, cap)

- A.coeff ← 새로운 배열(크기 : cap + 1)
- A.exp ← 새로운 배열(크기 : cap + 1)
- A.capacity ← cap
- A.size ← 0

INSERT(A, k, new_coeff, new_exp)

- 반복: i는 A.size부터 k까지 역순으로

  A.coeff[i + 1] ← A.coeff[i]

  A.exp[i + 1] ← A.exp[i]

- A.coeff[k] ← new_coeff
- A.exp[k] ← new_exp
- A.size ← A.size + 1

ADD(X, Y) 1/6

- A ← 새로운 항 리스트
- INITIALIZE(A, X.size + Y.size)
- iX ← 1, iY ← 1

- 반복: iX ≤ X.size 그리고 iY ≤ Y.size

ADD(X, Y) 2/6

만약: X.exp[iX] > Y.exp[iY]

INSERT(A, A.size + 1, X.coeff[iX], X.exp[iX])

iX ← iX + 1

다음 반복으로

ADD(X, Y) 3/6

만약: X.exp[iX] < Y.exp[iY]

INSERT(A, A.size + 1, Y.coeff[iY], Y.exp[iY])

iY ← iY + 1

다음 반복으로

ADD(X, Y) 4/6

만약: X.exp[iX] == Y.exp[iY]

new_coeff ← X.coeff[iX] + Y.coeff[iY]

INSERT(A, A.size + 1, new_coeff, X.exp[iX])

iX ← iX + 1

iY ← iY + 1

다음 반복으로

ADD(A, B) 5/6

- 만약: iX ≤ X.size

  반복: i는 iX부터 X.size까지

  INSERT(A, A.size + 1, X.coeff[iX], X.exp[iX])

ADD(A, B) 6/6

- 만약: iY ≤ Y.size

  반복: i는 iY부터 Y.size까지

  INSERT(A, A.size + 1, Y.coeff[iY], Y.exp[iY])

- 반환: A

요약

- 고정 크기 배열 리스트에서 데이터 추가/삭제
- 가변 크기 배열 리스트에서 데이터 추가
- 다항식 예제
- 다음 시간: 다항식 예제를 C 언어로 구현해 본다