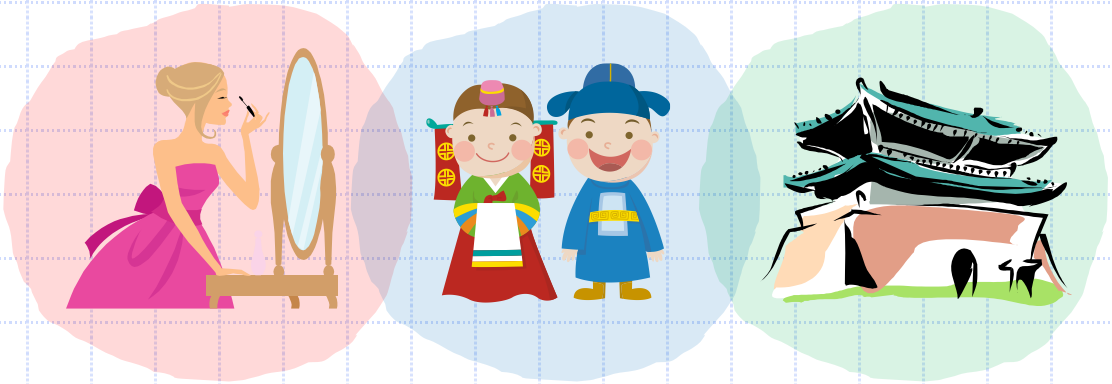
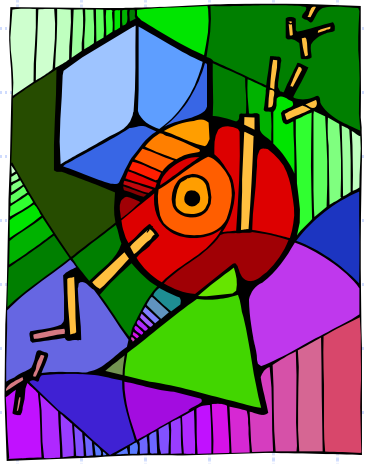
집합



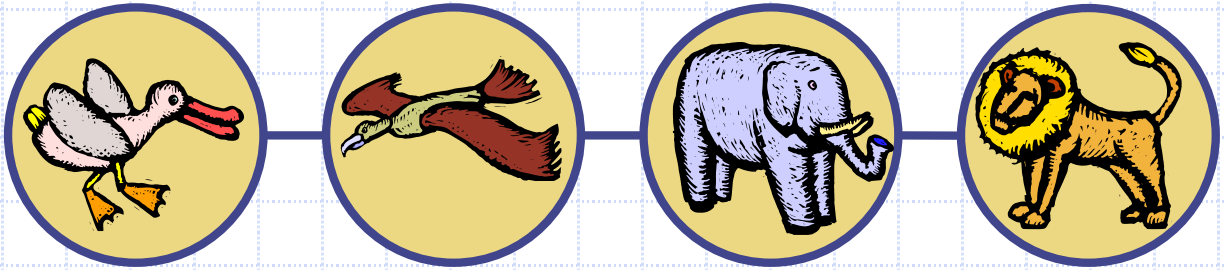
Outline

- 5.1 집합 ADT
- 5.2 집합 ADT 메쏘드
- 5.3 집합 ADT 구현
- 5.4 응용문제

집합 ADT

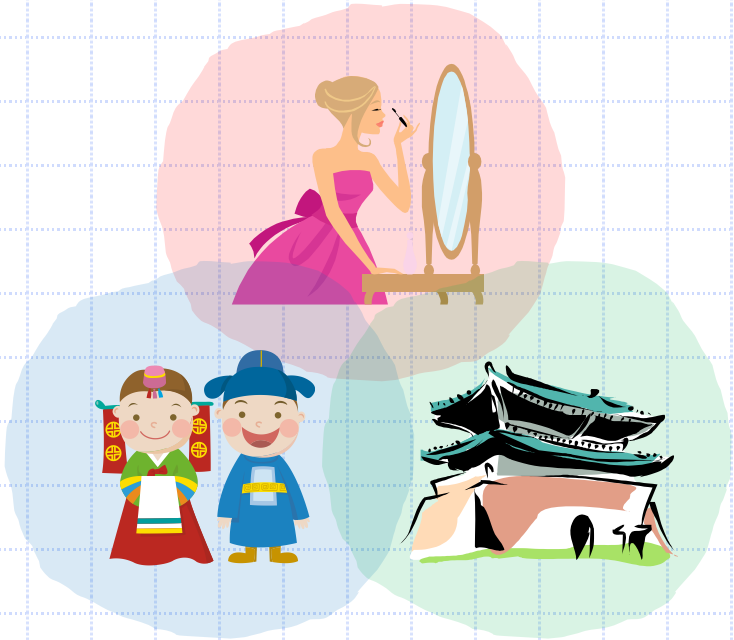


- 집합 ADT는 **유일한** 개체들을 담는 용기를 모델링한다
- 집합 ADT 관련 작업들의 효율적인 구현을 위해, 집합을 집합 원소들의 **정렬된** 리스트로 표현



집합 ADT 메쏘드

- 집합 A에 관한 주요 메쏘드
  - set union(B): 집합 B와의 **합집합**을 반환
  - set intersect(B): 집합 B와의 **교집합**을 반환
  - set subtract(B): 집합 B를 차감한 **차집합**을 반환
- 집합 A와 B에 관한 주요 작업의 실행시간은 최대 $O(|A| + |B|)$이 되어야 함



집합 ADT 메쏘드 (conti.)

- 일반 메쏘드
  - integer size()
  - boolean isEmpty()
  - iterator elements()
- 질의 메쏘드
  - boolean member(e): 개체 e가 집합의 원소인지 여부를 반환
  - boolean subset(B): 집합이 집합 B의 부분집합인지 여부를 반환
- 갱신 메쏘드
  - addElem(e): 집합에 원소 e를 추가
  - removeElem(e): 집합으로부터 원소 e를 삭제
- 예외
  - emptySetException(): 비어 있는 집합에 대해 삭제 혹은 첫 원소를 접근 시도할 경우 발령

집합 응용

- 직접 응용
  - 키워드 검색엔진
  - 집합론에 관련된 다양한 계산
- 간접 응용
  - 알고리즘을 위한 보조 데이터구조
  - 다른 데이터구조를 구성하는 요소

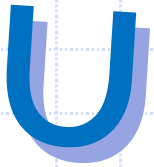
집합을 연결리스트에 저장

- 집합을 **연결리스트**로 구현 가능
  - 각 실제 노드는 하나의 집합원소를 표현
  - 효율을 위해 **헤더 및 트레일러 이중연결리스트**를 사용할 것을 추천
- 원소들을 일정한 순서에 의해 **정렬**하여 저장
- 기억장소 사용: $\mathbf{O}(n)$
- 대안으로, **배열**을 이용하여 집합을 저장 가능 (응용문제 참조)

집합 메쏘드 제시에 앞서

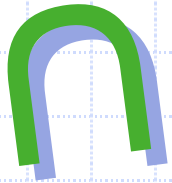
- 이어지는 합집합, 교집합, 차집합 메쏘드는:
  - **파괴적** – 즉, 대상 집합 A와 B를 보존하지 않는다
  - **참고:** 어떤 알고리즘이 **파괴적**이라 함은 알고리즘의 수행 결과 기존 데이터구조의 원형이 보존되지 않음을 의미(원형이 보존되는 **비파괴적** 버전의 반대 개념)
- **성능**
  - 모두 $\mathbf{O}(|A| + |B|)$ 시간에 수행
  - **전제:** addLast 작업을 $\mathbf{O}(1)$ 시간에 수행
    - 집합을 이중연결리스트로 구현함을 전제 – 단일연결리스트로 구현할 경우 집합 C의 마지막 노드 위치 관리 필요

합집합(union)



```
Alg union(B)
    input set A, B
    output set A∪B

1. C ← empty list
2. while (!A.isEmpty() & !B.isEmpty())
        a, b ← A.get(1), B.get(1)
        if (a < b)
            C.addLast(a)
            A.removeFirst()
        elseif (a > b)
            C. addLast(b)
            B.removeFirst()
        else {a = b}
            C.addLast(a)
            A.removeFirst()
            B.removeFirst()
3. while (!A.isEmpty())
        a ← A.get(1)
        C.addLast(a)
        A.removeFirst()
4. while (!B.isEmpty())
        b ← B.get(1)
        C.addLast(b)
        B.removeFirst()
5. return C
                                {Total O(|A| + |B|)}
```

교집합(intersect)



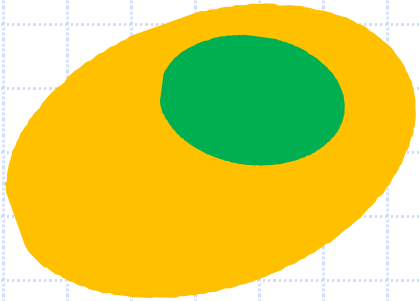
```
Alg intersect(B)
    input set A, B
    output set A∩B

1. C ← empty list
2. while (!A.isEmpty() & !B.isEmpty())
        a, b ← A.get(1), B.get(1)
        if (a < b)
            A.removeFirst()
        elseif (a > b)
            B.removeFirst()
        else {a = b}
            C.addLast(a)
            A.removeFirst()
            B.removeFirst()
3. while (!A.isEmpty())
        A.removeFirst()
4. while (!B.isEmpty())
        B.removeFirst()
5. return C
```

{Total **O**(|*A*| + |*B*|)}

차집합(subtract)

```
Alg subtract(B)
    input set A, B
    output set A – B

1. C ← empty list
2. while (!A.isEmpty() & !B.isEmpty())
        a, b ← A.get(1), B.get(1)
        if (a < b)
            C.addLast(a)
            A.removeFirst()
        elseif (a > b)
            B.removeFirst()
        else {a = b}
            A.removeFirst()
            B.removeFirst()
3. while (!A.isEmpty())
        a ← A.get(1)
        C.addLast(a)
        A.removeFirst()
4. while (!B.isEmpty())
        B.removeFirst()
5. return C
                                {Total O(|A| + |B|)}
```

소속과 부분집합

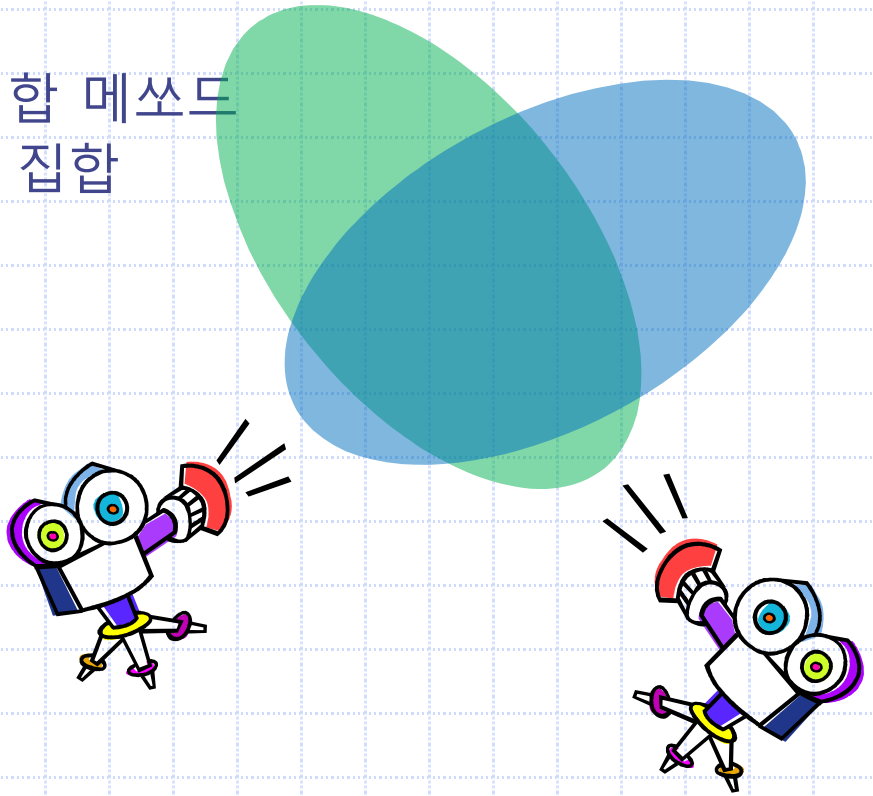


```
Alg member(e)
    input set A, element e
    output boolean

1. if (A = ∅)
        return False
2. p ← A
3. while (True)
        a ← p.elem
        if (a < e)
            if (p.next = ∅)   {p is last node}
                return False
            else
                p ← p.next
        elseif (a > e)
            return False
        else {a = e}
            return True
                                {Total O(|A|)}
```

```
Alg subset(B)
    input set A, B
    output boolean

1. if (A = ∅)
        return True
2. p ← A
3. while (True)
        if (B.member(p.elem))
            if (p.next = ∅)   {p is last node}
                return True
            else
                p ← p.next
        else
            return False

                                {Total O(|A||B|)}
```

응용문제

- 흥미로운 설계 문제를 통해 어떻게 **집합**을 주요 데이터구조로 사용하는지 공부
- 설계 문제
  - 비파괴적인 합집합, 교집합 메쏘드
  - 정수범위로 매핑 가능한 집합



응용문제: 비파괴적인 합집합, 교집합 메쏘드

- 주어진 두 개의 집합 A와 B에 대해, $A \cup B$와 $A \cap B$를 각각 계산하는 **재귀알고리즘** union(A, B)와 intersect(A, B)를 작성하라
- **전제**
  - 집합 A와 B는 각각 **정렬된 단일연결리스트**로 구현됨 – 여기서 A와 B는 각각 집합 A와 B의 첫 원소를 저장한 노드의 주소며, 공집합인 경우 주소는 널($\varnothing$)
  - 각 노드는 **elem**과 **next** 필드로 구성
  - 알고리즘 수행으로 인해 입력 집합 A 또는 B가 파괴되면 안 된다
- **주의:** 출력 집합은 **정렬된 단일연결리스트** 형태로 구축
- **힌트:** 비재귀버전에서는 필드 명에 대한 참조없이 일반 메쏘드만을 사용했던 것과 달리, 여기서는 **elem**과 **next** 필드를 직접 참조해도 무방

해결

```
Alg union(A, B)
    input set A, B
    output A∪B

1. if ((A = ∅) & (B = ∅))        {base case}
        return ∅
2. p ← getnode()
3. if (A = ∅)
        p.elem ← B.elem
        p.next ← union(A, B.next)
   elseif (B = ∅)
        p.elem ← A.elem
        p.next ← union(A.next, B)
   else
        if (A.elem < B.elem)
            p.elem ← A.elem
            p.next ← union(A.next, B)
        elseif (A.elem > B.elem)
            p.elem ← B.elem
            p.next ← union(A, B.next)
        else {A.elem = B.elem}
            p.elem ← A.elem
            p.next ← union(A.next, B.next)
3. return p
```

해결 (conti.)

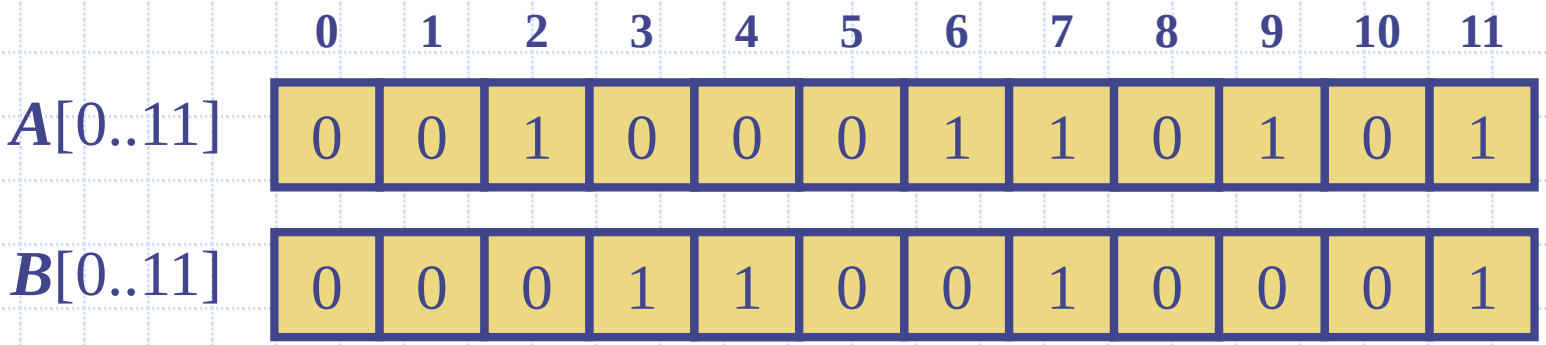
```
Alg intersect(A, B)
    input set A, B
    output A∩B

1. if ((A = ∅) || (B = ∅))    {base case}
        return ∅
2. if (A.elem < B.elem)
        return intersect(A.next, B)
    elseif (A.elem > B.elem)
        return intersect(A, B.next)
    else {A.elem = B.elem}
        p ← getnode()
        p.elem ← A.elem
        p.next ← intersect(A.next, B.next)
        return p
```

응용문제: 정수범위로 매핑 가능한 집합

- $[0, N-1]$ 범위의 정수 (또는 정수로 매핑 가능한) 원소들로 이루어진 집합이 있다
- **예**
  - 정수범위 = [2007, 2018]
    - ***Heat*** = {2009, 2013, 2014, 2016, 2018}
    - ***Dust*** = {2010, 2011, 2014, 2018}
  - 정수범위 = [months of the year]
    - ***Lee*** = {March, July, August, October, December}
    - ***Sung*** = {April, May, August, December}
- **문제:** 이러한 종류의 집합 S를 표현하는 효율적인 방안을 설계하라

해결: 비트벡터 사용

- 이런 종류의 집합 S를 표현하는 일반적인 방안: **비트벡터**(bitvector), 즉 논리벡터(boolean vector) V를 사용 – 여기서 "x가 S의 원소임"은 "$V[x] = True$"와 동치
- **예:** 정수범위 = $[0, 11]$
  - S_1 = {March, July, August, October, December}
  - S_2 = {April, May, August, December}



해결: 비트벡터 메쏘드

- 일반 메쏘드
  - integer size()
  - boolean isEmpty()
- 질의 메쏘드
  - boolean member(e): 개체 e가 집합의 원소인지 여부를 반환
  - boolean subset(B): 집합이 집합 B의 부분집합인지 여부를 반환
- 갱신 메쏘드
  - addElem(e): 집합에 원소 e를 추가
  - removeElem(e): 집합으로부터 원소 e를 삭제
- 예외
  - emptySetException(): 비어 있는 집합에 대해 삭제나 첫 원소 접근을 시도할 경우 발령