CSCI 1102 Computer Science 2

Spring 2021

Lecture Notes

Week 7: Priority Queues, Binary Heaps, Sets & Relations &Orders

Topics:

1. Priority Queues; Binary Heaps
2. Sets, Relations & Orders

---

1. Priority Queues; Binary Heaps

See the slides

2. Sets & Relations

The Idea

- Computer software is often required to keep track of "collections" of things.
- Mathematicians have thought carefully about these collections, and know them as *sets*.
- Software is also often required to keep track of the association between items from one set (the "keys") and another (the "values").

Preliminaries

Basic Sets

- A set is a collection of items with no duplicates.

  Examples: A = {1, 2, 3} B = {Bob, Alice, Joe} N = {0, 1, 2, ...} natural numbers

- NB: Only restriction on elements is identity.

Notation

alpha	beta	Gamma	gamma	delta	epsilon	lambda	sigma	tau
α	β	Γ	γ	δ	ϵ	λ	σ	τ

- A, B, C, ..., X, Y, Z for sets;
- $\emptyset$ or {} for the empty set;
- a, b, c, ... for elements of sets;
- $a \in A$ means that a is an element of set A;
- $(a_1, \ldots, a_n)$ is an n-tuple.

Variables and Quantifiers

- x, y, z for variables (which *vary* over sets!)
- $\forall x \in A$. *statement*

  asserts that *statement* holds for every element of A. For example, $\forall x \in \{1, 2, 3\}. x < 4$ means $1 < 4$ and $2 < 4$ and $3 < 4$. The occurrence of x adjacent to the quanitifer is called a *binding occurrence* of x; the occurrence of x to the right of the dot is called an *applied occurrence* or *use* of x. Note that we obtained the relation $1 < 4$ by plugging-in (or substituting) 1 for x in the statement $x < 4$.

- $\exists x \in A$. *statement*

  asserts that *statement* holds for some element of A.

Set Comprehensions

- { x | *statement* } means set of all x such that *statement* holds;

  Example

  Evens = $\{x \in \mathbb{N} \mid \exists y \in \mathbb{N} \text{ such that } x = 2y\}$

  or equivalently

  Evens = $\{x \mid x \in \mathbb{N} \text{ and } \exists y \in \mathbb{N} \text{ such that } x = 2y\}$

Basic Sets

- Notation:
  - Subset : $A \subseteq B$ means $\forall x \in A. x \in B$;
  - Set Equality : $A = B$ means $A \subseteq B$ and $B \subseteq A$.

Operations on Sets

- *Union* : $A_1 \cup \ldots \cup A_n = \{a \mid a \in A_i \text{ for some } i \in \{1, \ldots, n\}\}$;
- *Intersection* : $A_1 \cap \ldots \cap A_n = \{a \mid a \in A_i \text{ for every } i \in \{1, \ldots, n\}\}$;
- *Disjoint Union* : $A_1 + \ldots + A_n = \{(i, a) \mid a \in A_i \text{ for some } i \in \{1, \ldots, n\}\}$;
- *Product* : $A_1 \times \ldots \times A_n = \{(a_1, \ldots, a_n) \mid a_i \in A_i\}$;
- *Sequences* : Let A be a set and let ϵ denote the empty sequence.

  $A^* = \{w \mid w = \epsilon \text{ or } w = aw' \text{ with } a \in A \text{ and } w' \in A^*\}$

  Example: $\{a, b\}^* = \{\epsilon, a\epsilon, b\epsilon, aa\epsilon, ab\epsilon, \ldots\}$

Relations

- R is a(n n-ary) relation on sets $A_1, \ldots, A_n$ if $R \subseteq A_1 \times \ldots \times A_n$.
- When R is an n-ary relation on sets $A_1, \ldots, A_n$ and $A_1 = \ldots = A_n$ we say that R is an n-ary relation on A_1;
- When R is a finite set we say it is a finite relation.

Binary Relations

Let A and B be sets and let R be a relation on A, B.

Domain of Definition: DomDef(R) = $\{a \in A \mid \text{for some } b \in B, (a, b) \in R\}$

Example Relations

A = {1, 2, 3}, B = {Bob, Alice}

- R1 = A x B = {(1, Bob), (1, Alice), (2, Bob), (2, Alice), (3, Bob), (3, Alice)}
- R2 = {}
- R3 = {(1, Bob), (3, Alice)} // e.g., DomDef(R3) = {1, 3}
- R4 = {(1, Alice), (2, Alice), (3, Alice)}

Orders

Preorder

- Let R be a relation on A. R is *reflexive* iff $\forall x \in A. (x, x) \in R$;
- Let R be a relation on A. R is *transitive* iff $\forall x, y, z \in A.\ \text{If } (x, y) \in R \text{ and } (y, z) \in R \text{ then } (x, z) \in R.$
- A relation that is both reflexive and transitive is called a *preorder*.

Partial Orders

- Let R be a binary relation on A. R is *symmetric* iff $\forall x, y \in A.\ \text{if } (x, y) \in R \text{ then } (y, x) \in R$;
- Let R be as above. R is *antisymmetric* iff $\forall x, y \in A.\ \text{if } (x, y) \in R \text{ and } (y, x) \in R \text{ then } x = y$.
- A symmetric preorder is called an *equivalence relation*.
- An antisymmetric preorder is called a *partial order*.

Partially Ordered Sets

If R is a reflexive, antisymmetric and transitive binary relation on A, we say that

- R is a partial order on set A
- The set A is partially ordered by R
- A is a partially ordered set (not mentioning R)
- A is a poset

Notation

- If set A is partially ordered by R, we write (A, R) or more often $(A, \leq_R)$ or $(A, \leq)$ if R is implied by context;
- For $a, a' \in A$, instead of writing $(a, a') \in R$ we usually write $a \leq_R a'$ or $a \leq a'$ if R is implied.

- If $a \leq a'$ and $a! = a'$ we write $a < a'$.

Example

```
A = {Bob, Alice}

R5 = {(Bob, Bob), (Alice, Alice), (Bob, Alice)}
```

Hasse Diagram of a Relation

```
    Alice
      |
     Bob
```

Example

```
A = P({Bob, Alice, Joe}) = { {},
                             {Bob}, {Alice}, {Joe},
                             {Bob, Alice}, {Bob, Joe}, {Alice, Joe},
                             {Bob, Alice, Joe}
                           }
```

R6 = $(A, \subseteq)$ = {({}, {}), ({}, {Bob}), ({}, {Bob, Alice}), ...}

```
                               {Bob, Alice, Joe}
                              /      |        \
                     {Bob, Alice}  {Bob, Joe}   {Alice, Joe}
                          |     /\           /\     |
                        {Bob}      {Alice}       {Joe}
                               \      |        /
                                     {}
```

Total Order

- Let R be a partial order on A. R is a total order on A iff $\forall x, y \in A.$ either $(x, y) \in R$ or $(y, x) \in R$.
- Example : $(\mathbb{N}, \leq)$.

Lexicographic Ordering

Let A be a set and let $\leq_A$ be a partial order on A. We derive a partial order $\leq_{A^*}$ on A^* the sequences of elements from A.

$w \leq_{A^*} w'$ iff either $w = \epsilon$ or $w = av$, $w' = a'v'$ and either $a <_A a'$ or $a =_A a'$ and $v \leq_{A^*} v'$.

Example:

Let A = {p, q}. Then A* = {e, p, q, pq, ppq, … } and $pq \leq_{A^*} ppq$ because $a = p$, $v = q$, $a' = p$, $v' = pq$, $a = a'$ and $v \leq_{A^*} v'$ because $a = q$, $v = \epsilon$, $a' = p$, $v' = q$ and $v \leq_{A^*} v'$ because $v = \epsilon$.

Note: If $\leq$ is a partial order, then so is $\leq_{A^*}$.

Summary

In summary: we have type constructors: union, intersection, sum, product, sequence, -o-> and —>. Of these, sum, product, sequence, -o-> and —> have direct computational interpretations.