

ADT Taxonomy Part II

Note Title

9/28/2007

Plan for today

- Finish discussion of radix sort
- Briefly discuss bucket sort
- Return to ADT Taxonomy
- Introduce Set ADT
(if time permits)

Optimizing radix sort

time complexity $C \cdot \frac{b}{r} (n + 2^r)$

bits \rightarrow b

base for a r -bit digit \rightarrow r

bits/digit \rightarrow $\frac{b}{r}$

intuitively want to set r so $n = 2^r$ so you reduce # digits while not making base too high

Solving for r in $n = 2^r$ yields $r = \log_2 n$

Example: let $c=5$, $n=1,000,000$
 32 bit numbers

| | <u># bits / digit</u> | <u># digits</u> | <u>base^k</u> | <u>$c \cdot d(n+k)$</u> |
|------------------------|-----------------------|-----------------|-------------------------|------------------------------------|
| basic radix sort | 1 | 32 | 2 | 160,000,300 |
| | 2 | 16 | 4 | 80,000,320 |
| | 4 | 8 | 16 | 40,000,640 |
| | 8 | 4 | 256 | 20,005,120 |
| | 16 | 2 | 65536 | 10,655,360 |
| Counting Sort | 32 | 1 | $> 4 \times 10^9$ | $> 4,000,000,000$ |

What are the limitations of radix sort?

Requires that you can digitize all elements.

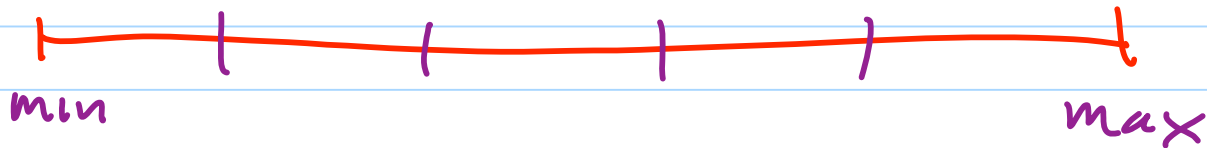
Instead of using a comparator to compare entire elements, a digitizer is used to extract each digit.

Think about these costs.

Bucket sort

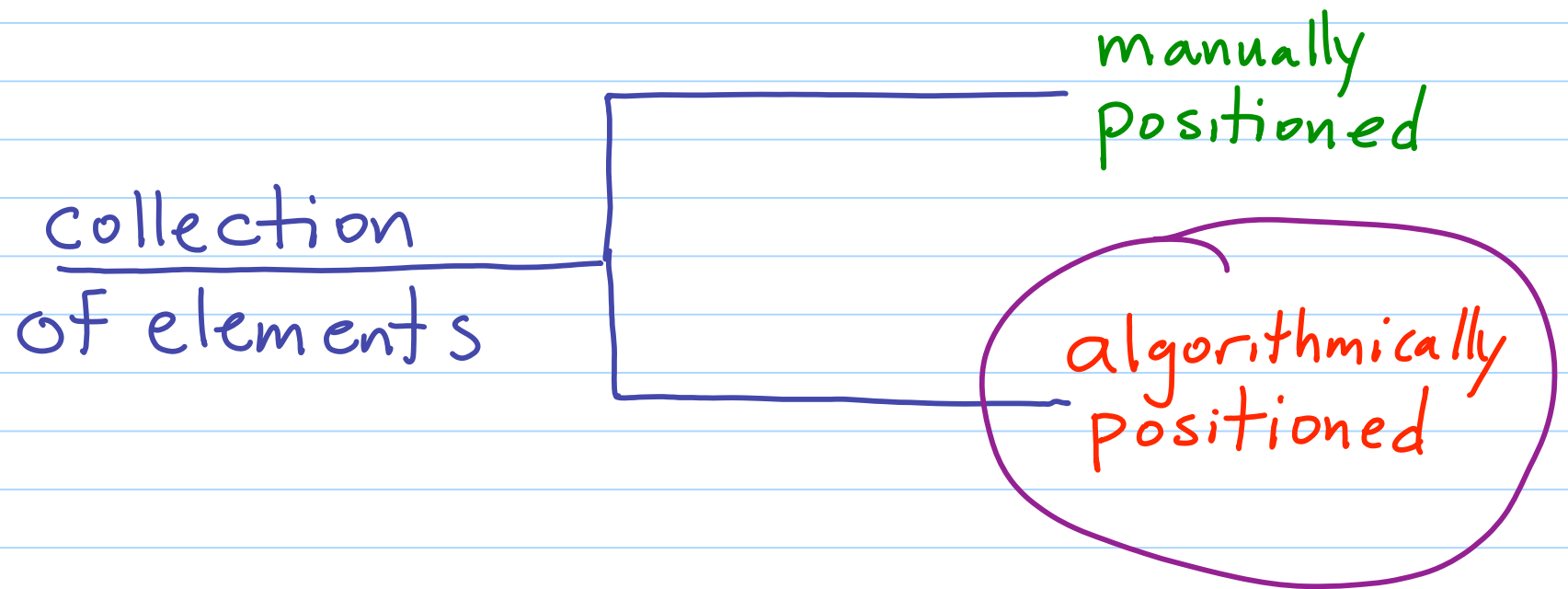
Define Bucketizer that divides the range of all (n) elements into n buckets

$n = 5$

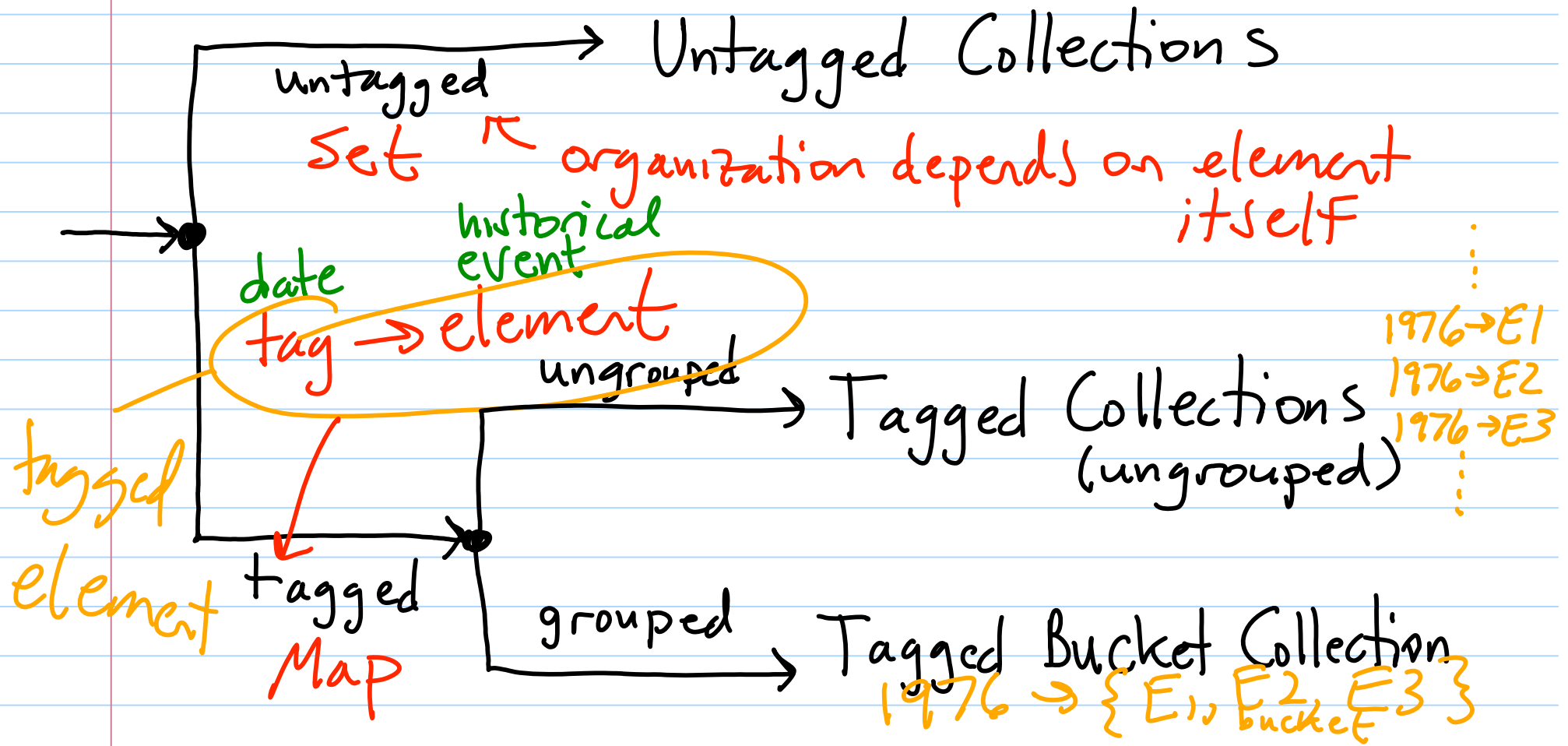


$O(n)$ Place each element in correct bucket
Insertion sort to finish up

ADT Taxonomy, Part II

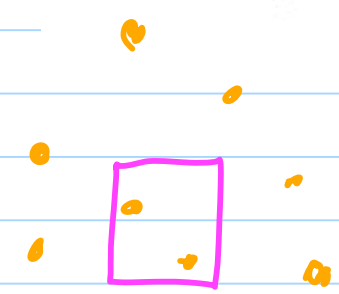
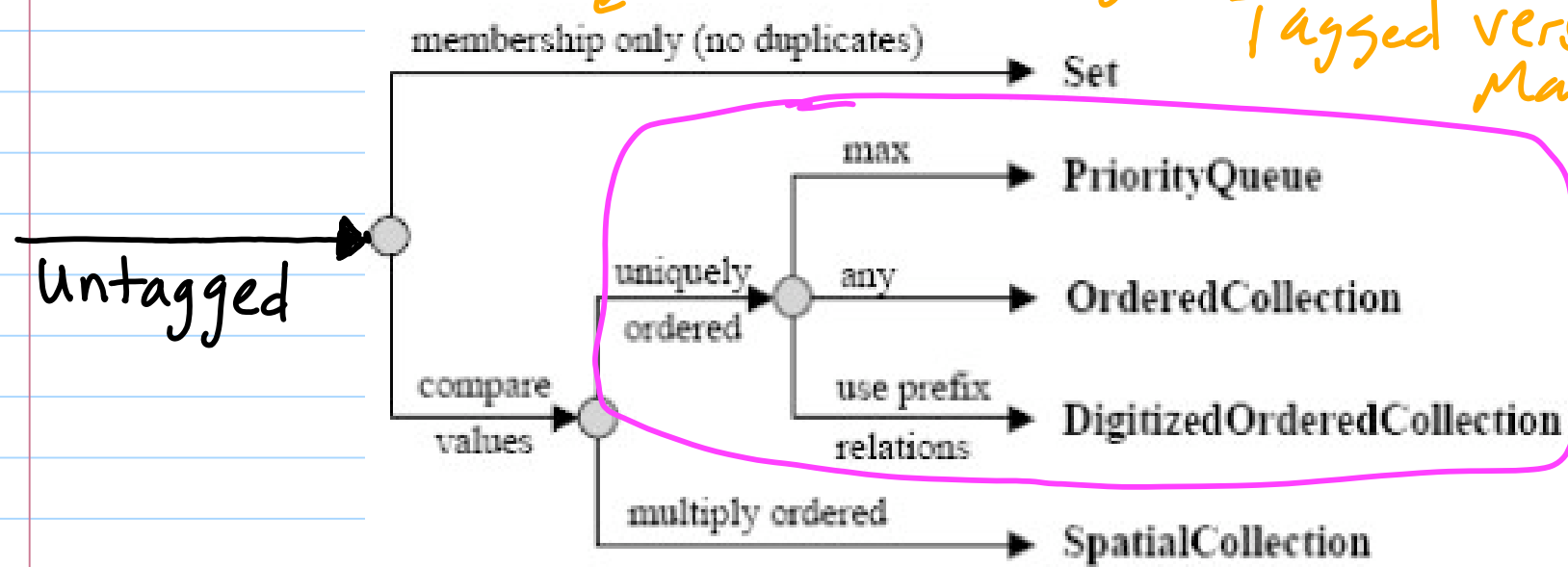


Algorithmically Positioned Collection

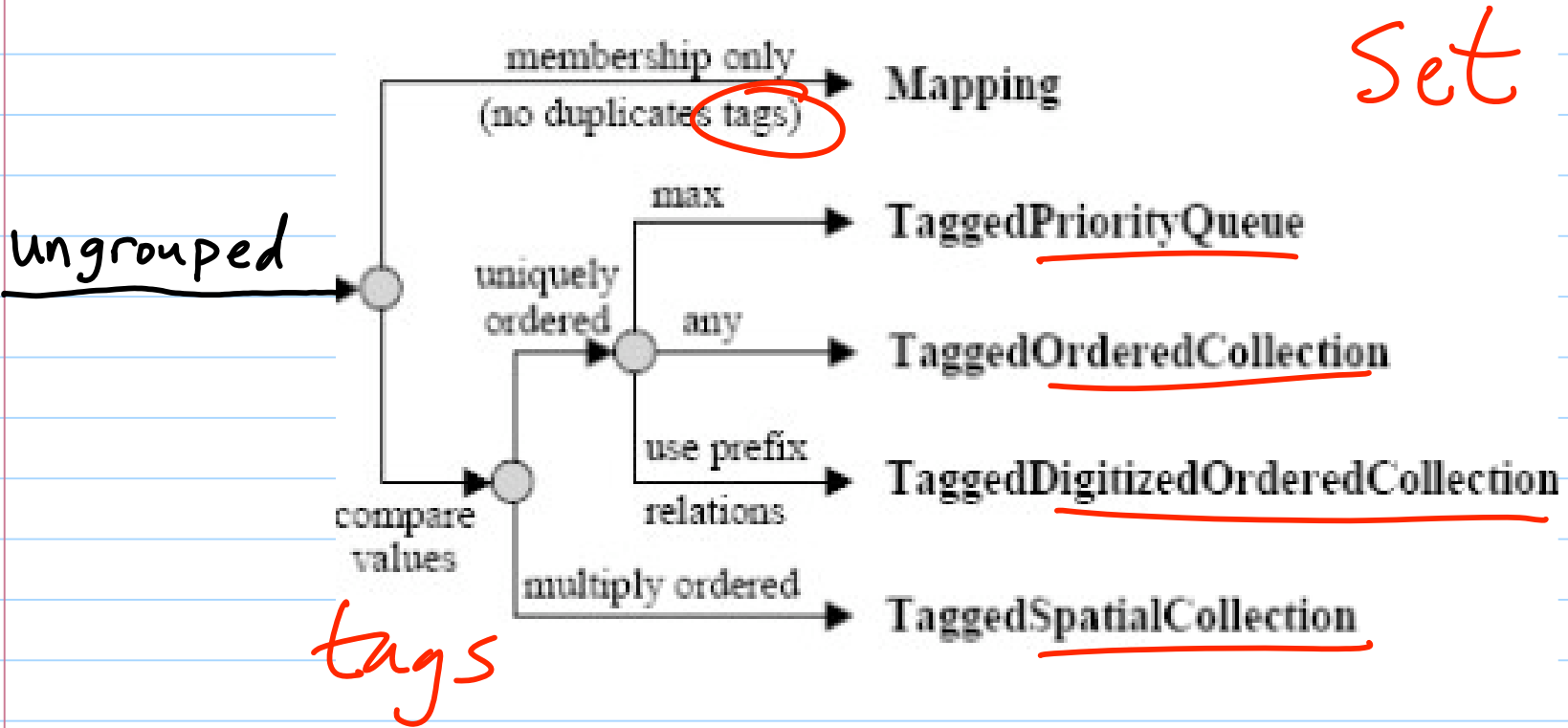


Untagged Algorithmically Positioned Collection

↙ notion of equivalence used
Tagged version Mapping

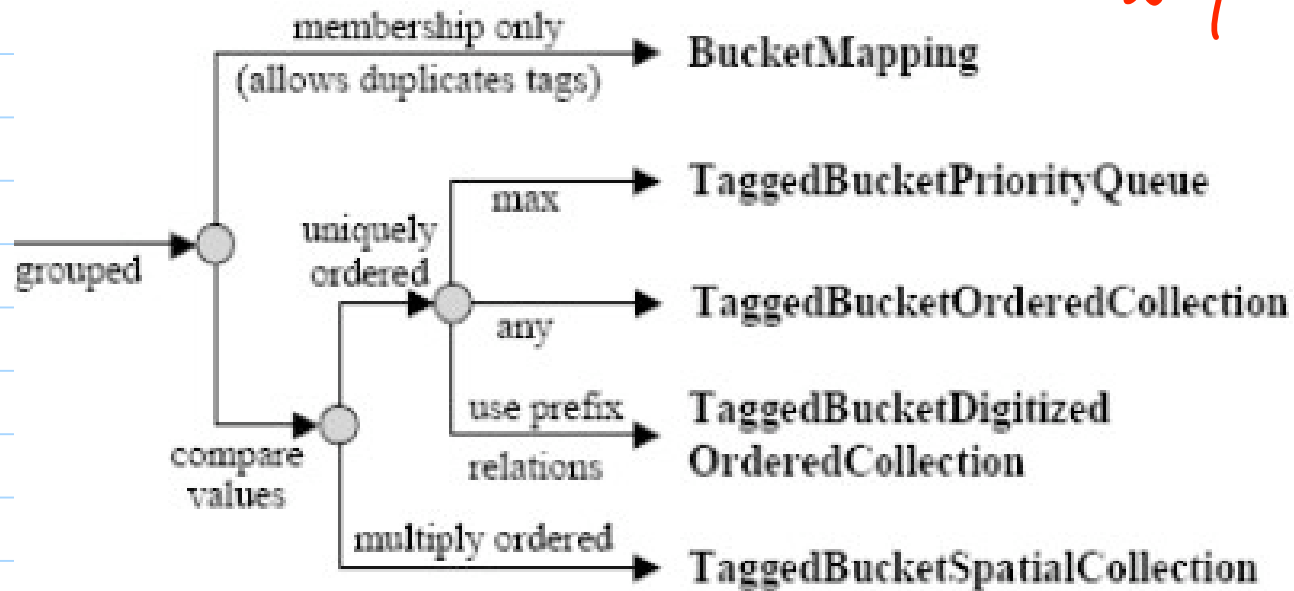


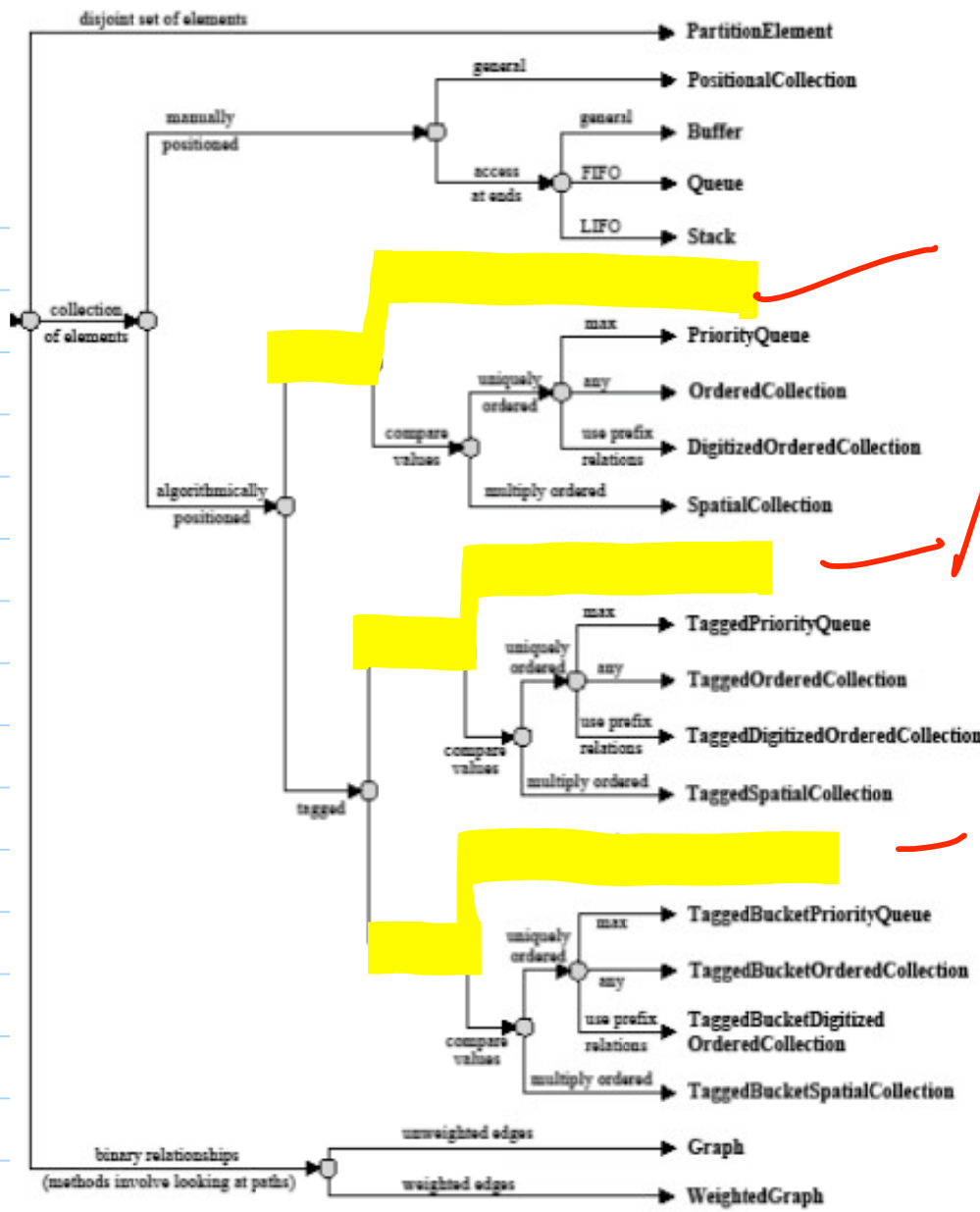
tagged element { tag | element } insert into an untagged collection
 Tagged Ungrouped Algorithmically Positioned Collection



Tagged Grouped Algorithmically Positioned Collection

tag, bucket
↙ any data structure





Set

Mapping

Bucket Mapping

Set ADT

wrap a tagged element — mapping
collection ADT

wrap a tag → bucket of element — Bucket
mapping ADT

Fundamental Methods to Support

insert (add, put)
find (tag, e)
remove

Set

add(e)

contains(e)

equivalence depends
on element

Mapping or
Bucket Mapping

put(tag, e)

contains(tag)

equivalence depends
on tag