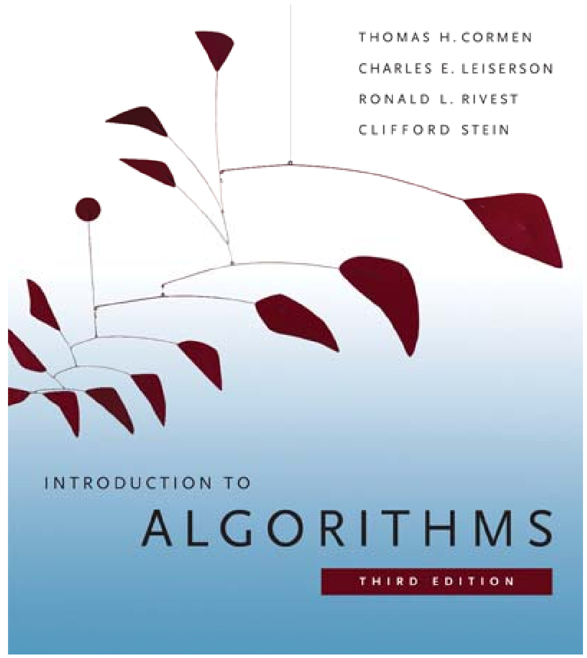
6.006- *Introduction to Algorithms*



Lecture 7

Prof. Manolis Kellis

CLRS: 11.4, 17.

Unit #2 – Genomes, Hashing, and Dictionaries

Unit	Pset	Week	Date		Lecture (Tuesdays and Thursdays)		Recitation (Wed and Fri)
Intro	PS1	1	Tue Feb 01	1	Introduction and Document Distance	1	Python and Asymptotic Complexity
Binary	Out: 2/1		Thu Feb 03	2	Peak Finding Problem	2	Peak Finding correctness & analysis
Search	Due: Mon 2/14	2	Tue Feb 08	3	Scheduling and Binary Search Trees	3	Binary Search Tree Operations
Trees	HW lab: Sun 2/13		Thu Feb 10	4	Balanced Binary Search Trees	4	Rotations and AVL tree deletions
Hashing	PS2 Out: 2/15	3	Tue Feb 15	5	Hashing I : Chaining, Hash Functions	5	Hash recipes, collisions, Python dicts
	Due: Mon 2/28		Thu Feb 17	6	Hashing II : Table Doubling, Rolling Hash	6	Probability review, Pattern matching
	HW lab:Sun 2/27	4	Tue Feb 22	-	President's Day - Monday Schedule - No Class	-	No recitation
			Thu Feb 24	7	Hashing III : Open Addressing	7	Universal Hashing, Perfect Hashing
Sorting	PS3. Out: 3/1	5	Tue Mar 01	8	Sorting I : Insertion & Merge Sort, Master Theorem	8	Proof of Master Theorem, Examples
	Due: Mon 3/7		Thu Mar 03	9	Sorting II : Heaps	9	Heap Operations
	HW lab: Sun 3/6	6	Tue Mar 08	10	Sorting III: Lower Bounds, Counting Sort, Radix Sort	10	Models of computation
			Wed Mar 09	Q1	Quiz 1 in class at 7:30pm. Covers L1-R10. Review Session on Tue 3/8 at 7:30pm.		
Graphs	PS4. Out: 3/10		Thu Mar 10	11	Searching I: Graph Representation, Depth-1st Search	11	Strongly connected components
and	Due: Fri 3/18	7	Tue Mar 15	12	Searching II: Breadth-1st Search, Topological Sort	12	Rubik's Cube Solving
Search	HW lab:W 3/16		Thu Mar 17	13	Searching III: Games, Network properties, Motifs	13	Subgraph isomorphism
Shortest	PS5	8	Tue Mar 29	14	Shortest Paths I: Introduction, Bellman-Ford	14	Relaxation algorithms
Paths	Out: 3/29		Thu Mar 31	15	Shortest Paths II: Bellman-Ford, DAGs	15	Shortest Path applications
	Due: Mon 4/11	9	Tue Apr 05	16	Shortest Paths III: Dijkstra	16	Speeding up Dijkstra's algorithm
	HW lab:Sun 4/10		Thu Apr 07	17	Graph applications, Genome Assembly	17	Euler Tours
Dynamic	PS6	10	Tue Apr 12	18	DP I: Memoization, Fibonacci, Crazy Eights	18	Limits of dynamic programming
Program	Out: Tue 4/12		Wed Apr 13	Q2	Quiz 2 in class at 7:30pm. Covers L11-R17. Review Session on Tue 4/13 at 7:30pm.		
ming	Due: Fri 4/29		Thu Apr 14	19	DP II: Shortest Paths, Genome sequence alignment	19	Edit Distance, LCS, cost functions
	HW lab:W 4/27	11	Tue Apr 19	-	Patriot's Day - Monday and Tuesday Off	-	No recitation
			Thu Apr 21	20	DP III: Text Justification, Knapsack	20	Saving Princess Peach
		12	Tue Apr 26	21	DP IV: Piano Fingering, Vertex Cover, Structured DP	21	Phylogeny
Numbers	PS7 out Thu4/28		Thu Apr 28	22	Numerics I - Computing on large numbers	22	Models of computation return!
Pictures	Due: Fri 5/6	13	Tue May 3	23	Numerics II - Iterative algorithms, Newton's method	23	Computing the nth digit of π
(NP)	HW lab: Wed 5/4		Thu May 5	24	Geometry: Line sweep, Convex Hull	24	Closest pair
		14	Tue May 10	25	Complexity classes, and reductions	25	Undecidability of Life
Beyond			Thu May 12	26	Research Directions (15 mins each) + related classes		
		15	Finals week	Q3	Final exam is cumulative L1-L26. Emphasis on L18-L26. Review Session on Fri 5/13 at 3pm		

Unit #2: Hashing

- **Last Tues: Genomes, Dictionaries, Hashing**
  - – Intro, basic operations, collisions and chaining
  - – Simple uniform hashing assumption
- **Last Thur: Faster hashing, hash functions**
  - – Hash functions in practice: div/mult/python
  - – Faster hashing: Rolling Hash: $O(n^2) \rightarrow O(n \lg n)$
  - – Faster comparison: Signatures: mismatch time
- **Today: Space issues**
  - – Dynamic resizing and amortized analysis
  - – Open addressing, deletions, and probing
  - – Advanced topics: universal hashing, fingerprints

Today: Hashing III: Space issues

Rev: Hash functs, chaining, SUHA, rolling, signatures

- **Dynamic dictionaries:** Resizing hash tables
  - When to resize: insertions, deletions
  - Resizing operations, amortized analysis
- **Open addressing:** Doing away w/ linked lists
  - Operations: insertion, deletion
  - Probing: linear probing, double hashing
  - Performance analysis: UHA, open vs. chaining
- **Advanced topics:** Randomized algorithms (shht!)
  - Universal hashing, perfect hashing
  - Fingerprinting, file signatures, false positives

Remember Hashing I and II

- Hashing and hash functions
  - Humongous universe of keys → itty bitty little space
- Hash table as dictionary
  - Insert/Search/Delete
- Collisions by chaining
  - Build a linked list in each bucket
  - Operation time is length of list
- Simple Uniform Hashing Assumption
  - Every item to uniform random bucket
  - n items in size m table → average length n/m = α
- Speeding up hashing
  - Rolling Hash: fast sequence of hash's
  - Signatures: fast comparison, avoid frequent mismatches
- Comparing genomes
  - $O(n^4) \rightarrow O_{binsearchL}(n^3 \lg n) \rightarrow O_{hash}(n^2 \lg n) \rightarrow O_{roll/sign}(n \lg n)$

Today: Hashing III: Space issues

Rev: Hash functs, chaining, SUHA, rolling, signatures

- **Dynamic dictionaries:** Resizing hash tables
  - When to resize: insertions, deletions
  - Resizing operations, amortized analysis
- **Open addressing:** Doing away w/ linked lists
  - Operations: insertion, deletion
  - Probing: linear probing, double hashing
  - Performance analysis: UHA, open vs. chaining
- **Advanced topics:** Randomized algorithms (shht!)
  - Universal hashing, perfect hashing
  - Fingerprinting, file signatures, false positives

Dynamic Dictionaries

- In substring application, inserted all at once then scanned
- More generally, arbitrary sequence of insert, delete, find
- How do we know how big the table will get?
- What if we guess wrong?

  too small ➔ load high, operations too slow

  too large ➔ high initialization cost, consumes space, potentially more cache-misses
- Want $m=\Theta(n)$ at all times

Solution: Resize when needed

- Start table at small constant size
- When table too full, make it bigger
- When table too empty, make it smaller
- How?
  - Build a whole new hash table and insert items
  - Pick new hash ‘seed’, recompute all hashes
  - Recreate new linked lists
  - Time spent to rebuild:

    (new-size) + #hashes x (HashTime)

When to resize?

- **Approach 1:** whenever $n > m$, rebuild table to new size
  - Sequence of n inserts
  - Each increases n past m, causes rebuild
  - Total work: $\Theta(1 + 2 + \ldots + n) = \Theta(n^2)$
- **Approach 2:** Whenever $n \geq 2m$, rebuild table to new size
  - ***Costly inserts*:** insert 2^i for all i:
    These cost: $\Theta(1 + 2 + 4 + \ldots + n) = \Theta(n)$
  - All other inserts take O(1) time – ***why?***
  - Inserting n items takes O(n) time
  - Keeps m a power of 2 --- good for mod

a factor of (HashTime) is suppressed here

Amortized Analysis

- If a sequence of n operations takes time T, then each operation has amortized cost T/n
  - Like amortizing a loan: payment per month
- Rebuilding when $n \geq 2m \rightarrow$ some ops are very slow
  - $\Theta(n)$ for insertion that causes last resize
- But on average, fast
  - O(1) amortized cost per operation
- Often, only care about total runtime
  - So averaging is fine

Insertions+Deletions?

- Rebuild table to new size when n < m?
  - Same as bad insert: $O(n^2)$ work
- Rebuild when n<m/2?
  - Makes a sequence of deletes fast
  - What about an arbitrary sequence of inserts/deletes?
    - Suppose we have just rebuilt: m=n
    - Next rebuild a grow → at least m more inserts are needed before growing table
      - Amortized cost O(2m / m)) = O(1)

        Cost to rebuild

        Paid after m insertions
    - Next rebuild a shrink → at least m/2 more deletes are needed before shrinking
      - Amortized cost O(m/2 / (m/2)) = O(1)

        Cost to rebuild

        Paid after m/2 deletions

Putting the two together

- Algorithm
  - Keep m a power of 2 (good for mod)
  - Grow (double m) when $n \geq m$
  - Shrink (halve m) when $n \leq m/4$
- Analysis
  - Just after rebuild: $n=m/2$
  - Next rebuild a grow → at least m/2 more inserts
    - Amortized cost $O(2m / (m/2)) = O(1)$
  - Next rebuild a shrink → at least m/4 more deletes
    - Amortized cost $O(m/2 / (m/4)) = O(1)$

Summary

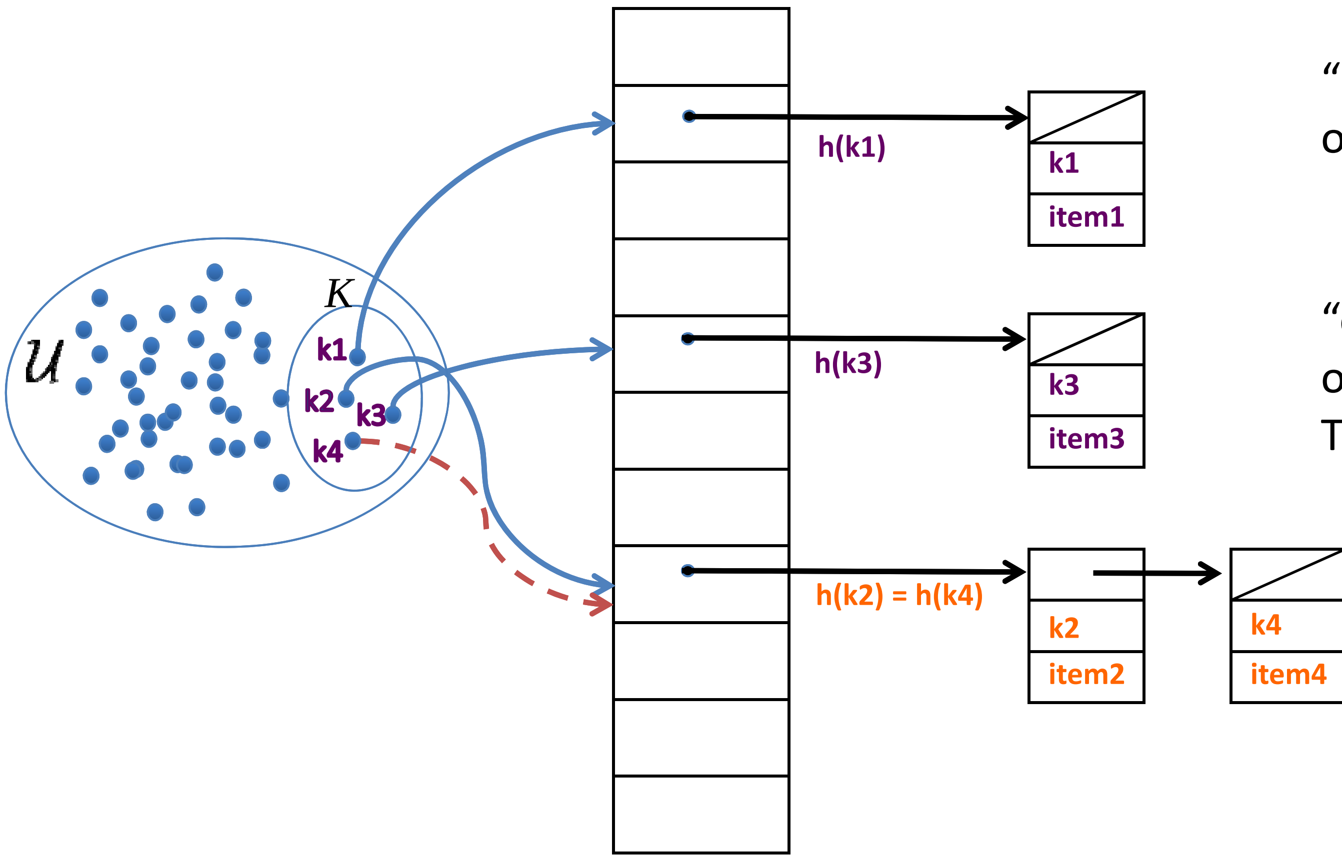
- Arbitrary sequence of insert/delete/find
- O(1) amortized time per operation

Today: Hashing III: Space issues

Rev: Hash functs, chaining, SUHA, rolling, signatures

- ✓ **Dynamic dictionaries:** Resizing hash tables
  - ✓ When to resize: insertions, deletions
  - ✓ Resizing operations, amortized analysis
- ➢ **Open addressing:** Doing away w/ linked lists
  - ❑ Operations: insertion, deletion
  - ❑ Probing: linear probing, double hashing
  - ❑ Performance analysis: UHA, open vs. chaining
- ❑ **Advanced topics:** Randomized algorithms (shht!)
  - ❑ Universal hashing, perfect hashing
  - ❑ Fingerprinting, file signatures, false positives

The trouble with chaining



“my heart is sick
of being in chains”
-- Tori Amos ‘92

“our people are sick
of being in chains”
Tunisia, Egypt, Libya ‘10

- Hash table just for indexing, all storage in linked lists
- In practice: Bad locality of reference for table items
- Would like to store only table in memory, with all items

Open Addressing

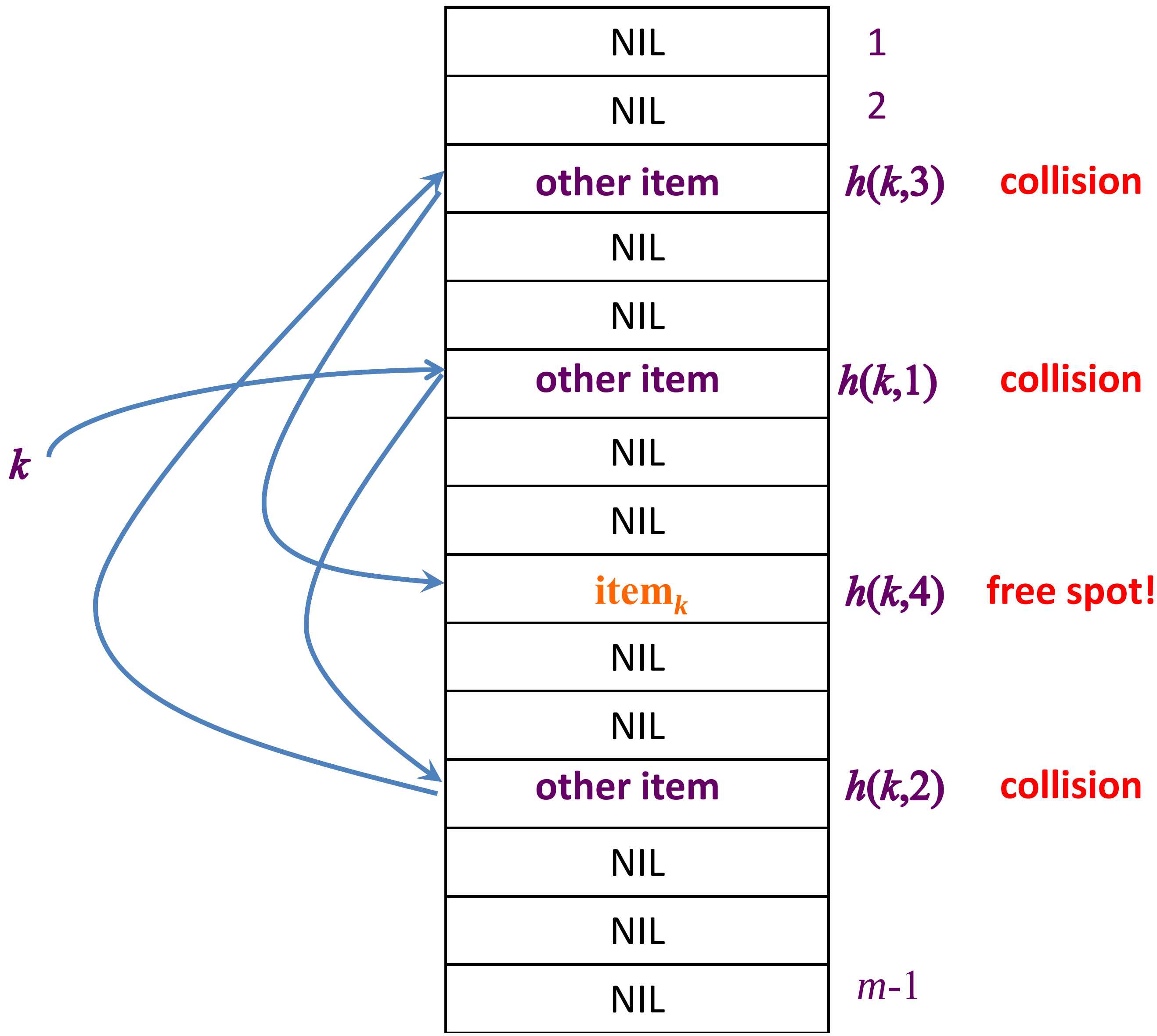
- Different technique for dealing with collisions; does not use linked list
- Instead: if bucket occupied, find other bucket (need $m \geq n$)
- For insert: probe a sequence of buckets until find empty one!
- $h(x)$ specifies probe sequence for item x
  - Ideally, sequence visits all buckets
  - h: U ´ [1..m] → [1..m]

Bucket

Universe of keys

Probe number

Open Addressing (example)



Operations

- Insert
  - Probe till find empty bucket, put item there
- Search
  - Probe till find item (return with success)
  - Or find empty bucket (return with failure)
    - Because if item inserted, would use that empty bucket
- Delete
  - Probe till find item
  - Remove, ~~leaving empty~~ bucket (NIL)

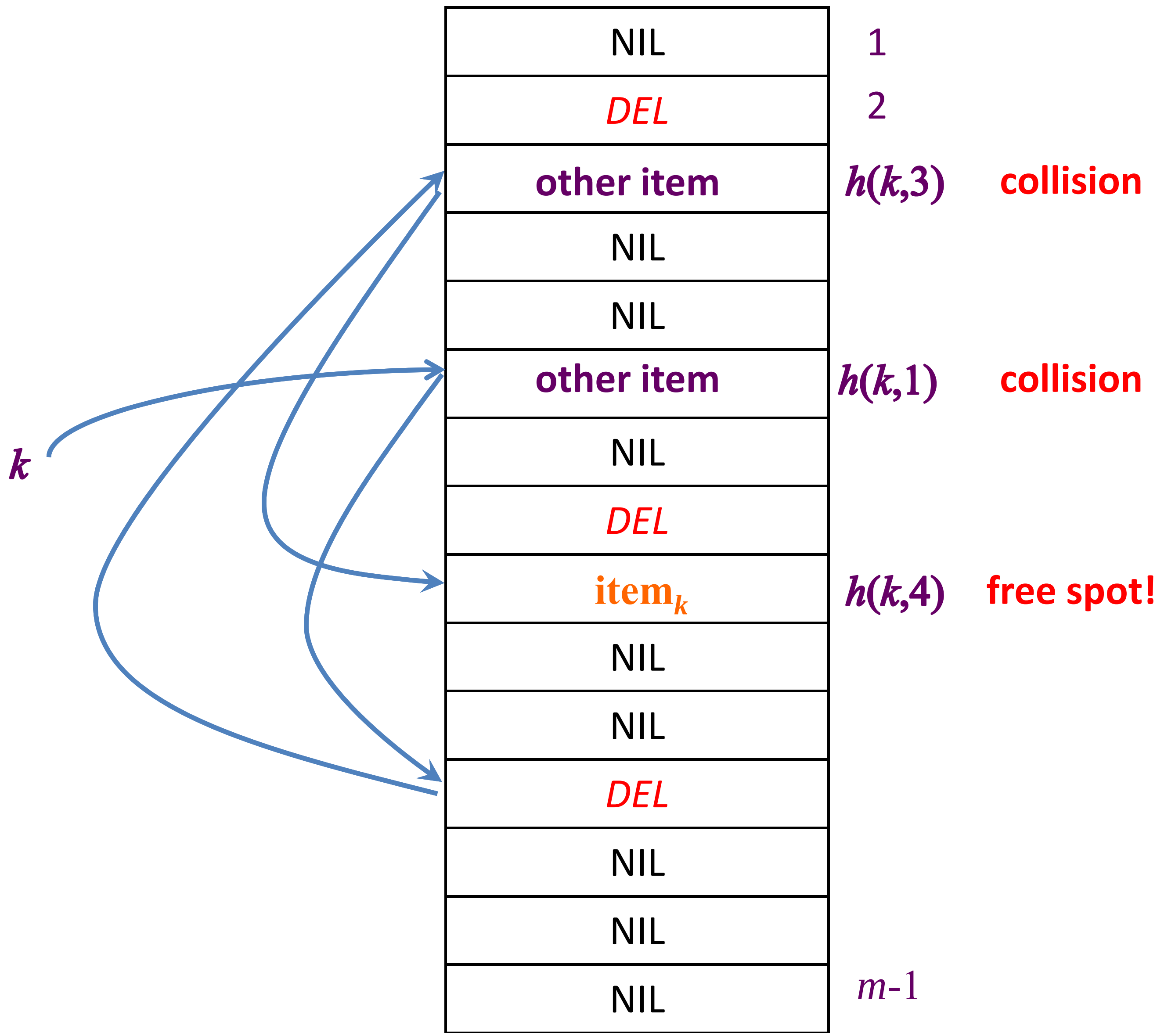
Problem with Deletion

- Consider a sequence
  - Insert x
  - Insert y
    - suppose probe sequence for y passes x bucket
    - store y elsewhere
  - Delete x (leaving hole)
  - Search for y
    - Probe sequence hits x bucket
    - Bucket now empty
    - Conclude y not in table (else y would be there)

Solution for deletion

- When delete x
  - Leave it in bucket
  - But mark it deleted --- store “tombstone” (*DEL*)
- Future search for x sees x is deleted
  - Returns “x not found”
- “Insert z” probes may hit x bucket
  - Since x is deleted, overwrite with z
  - So keeping deleted items doesn’t waste space

Open Addressing (example after del 2)



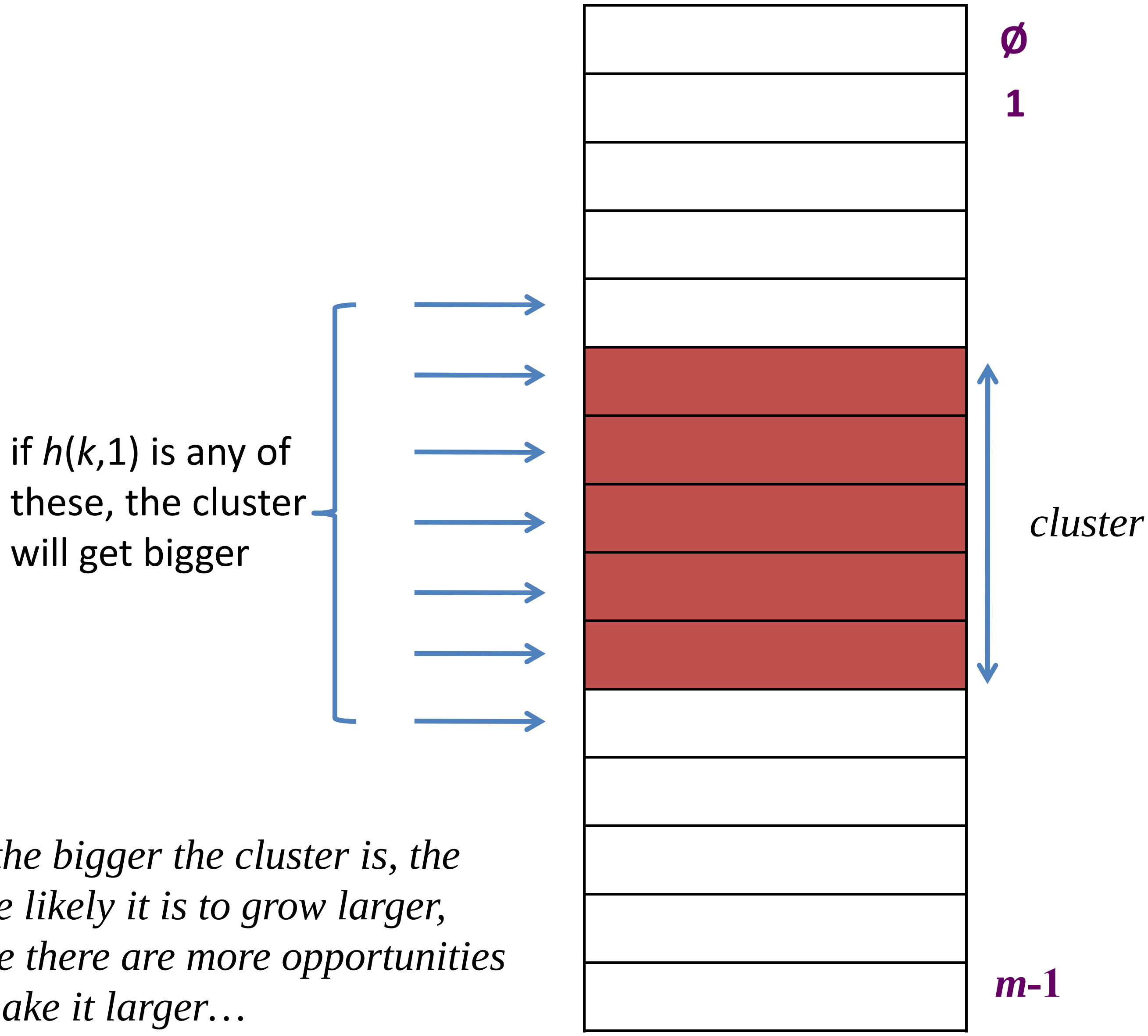
Today: Hashing III: Space issues

Rev: Hash functs, chaining, SUHA, rolling, signatures

- ✓ **Dynamic dictionaries:** Resizing hash tables
  - ✓ When to resize: insertions, deletions
  - ✓ Resizing operations, amortized analysis
- ✓ **Open addressing:** Doing away w/ linked lists
  - ✓ Operations: insertion, deletion
  - ➢ Probing: linear probing, double hashing
  - ❑ Performance analysis: UHA, open vs. chaining
- ❑ **Advanced topics:** Randomized algorithms (shht!)
  - ❑ Universal hashing, perfect hashing
  - ❑ Fingerprinting, file signatures, false positives

Linear probing

- h(k,i) = h’(k) + i for ordinary hash h’
- Problem: creates “clusters”, i.e. sequences of full buckets
  - exactly like parking
  - Big clusters are hit by lots of new items
  - They get put at end of cluster
  - Big cluster gets bigger: “rich get richer” phenomenon



i.e. the bigger the cluster is, the more likely it is to grow larger, since there are more opportunities to make it larger…

Linear probing

- $h(k,i) = h'(k) + i$ for ordinary hash h'
- Problem: creates "clusters", i.e. sequences of full buckets
  - exactly like parking
  - Big clusters are hit by lots of new items
  - They get put at end of cluster
  - Big cluster gets bigger: "rich get richer" phenomenon
- For $0.1 < \alpha < 0.99$, cluster size $\Theta(\log n)$
- Wrecks our constant-time operations

Double Hashing

- Two ordinary hash functions f(k), g(k)
- Probe sequence $h(k,i) = f(k) + i \cdot g(k) \bmod m$
- If g(k) relatively prime to m, hits all buckets
  - E.g., if $m=2^r$, make g(k) odd
  - The same bucket is hit twice if for some i,j:

    $f(k) + i \cdot g(k) = f(k) + j \cdot g(k) \bmod m$

    $\rightarrow i \cdot g(k) = j \cdot g(k) \pmod m$

    $\rightarrow (i-j) \cdot g(k) = 0 \pmod m$

    $\rightarrow$ m and g(k) not relatively prime

(otherwise m should divide i-j, which is not possible for $i, j<m$)

Today: Hashing III: Space issues

Rev: Hash functs, chaining, SUHA, rolling, signatures

- ✓ **Dynamic dictionaries:** Resizing hash tables
  - ✓ When to resize: insertions, deletions
  - ✓ Resizing operations, amortized analysis
- ✓ **Open addressing:** Doing away w/ linked lists
  - ✓ Operations: insertion, deletion
  - ✓ Probing: linear probing, double hashing
  - ➢ Performance analysis: UHA, open vs. chaining
- ❑ **Advanced topics:** Randomized algorithms (shht!)
  - ❑ Universal hashing, perfect hashing
  - ❑ Fingerprinting, file signatures, false positives

Performance of Open Addressing

- Operation time is length of probe sequence
- How long is it?
- In general, hard to answer.
- Introducing…
- "Uniform Hashing Assumption" (UHA):
  - Probe sequence is a uniform random permutation of [1..m]
  - (N.B. this is different to the simple uniform hashing assumption (SUHA))

Analysis under UHA

- Suppose:
  - a size-m table contains n items
  - we are using open addressing
  - we are about to insert new item
- Probability first probe successful? P(free slot)

Free slots → **Total slots** →

$$\frac{m-n}{m} := p$$

Why? From UHA, probe sequence random permutation
Hence, first position probed random
m-n out of the m slots are unoccupied

Analysis under UHA: 2nd probe

- If first probe unsuccessful, probability second prob successful?

Free slots → numerator; **Total slots** → denominator

$$\frac{m-n}{m-1} \geq \frac{m-n}{m} = p$$

Why?

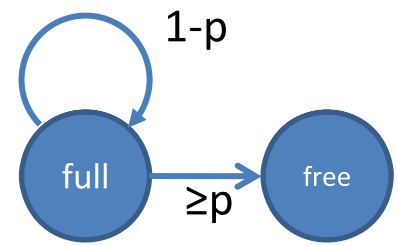
- From UHA, probe sequence random permutation
- Hence, first probed slot is random; the second probed slot is random among the remaining slots, etc.
- Since first probe unsuccessful, it probed an occupied slot
- Hence, the second probe is choosing uniformly from m-1 slots, among which m-n are still clean

Analysis under UHA: 3rd probe

- If first two probes unsuccessful, probability third prob successful?

Free slots →
Total slots →

$$\underbrace{\frac{m-n}{m-2}}_{\geq p} \geq \underbrace{\frac{m-n}{m}}_{\geq p} = p$$



- …

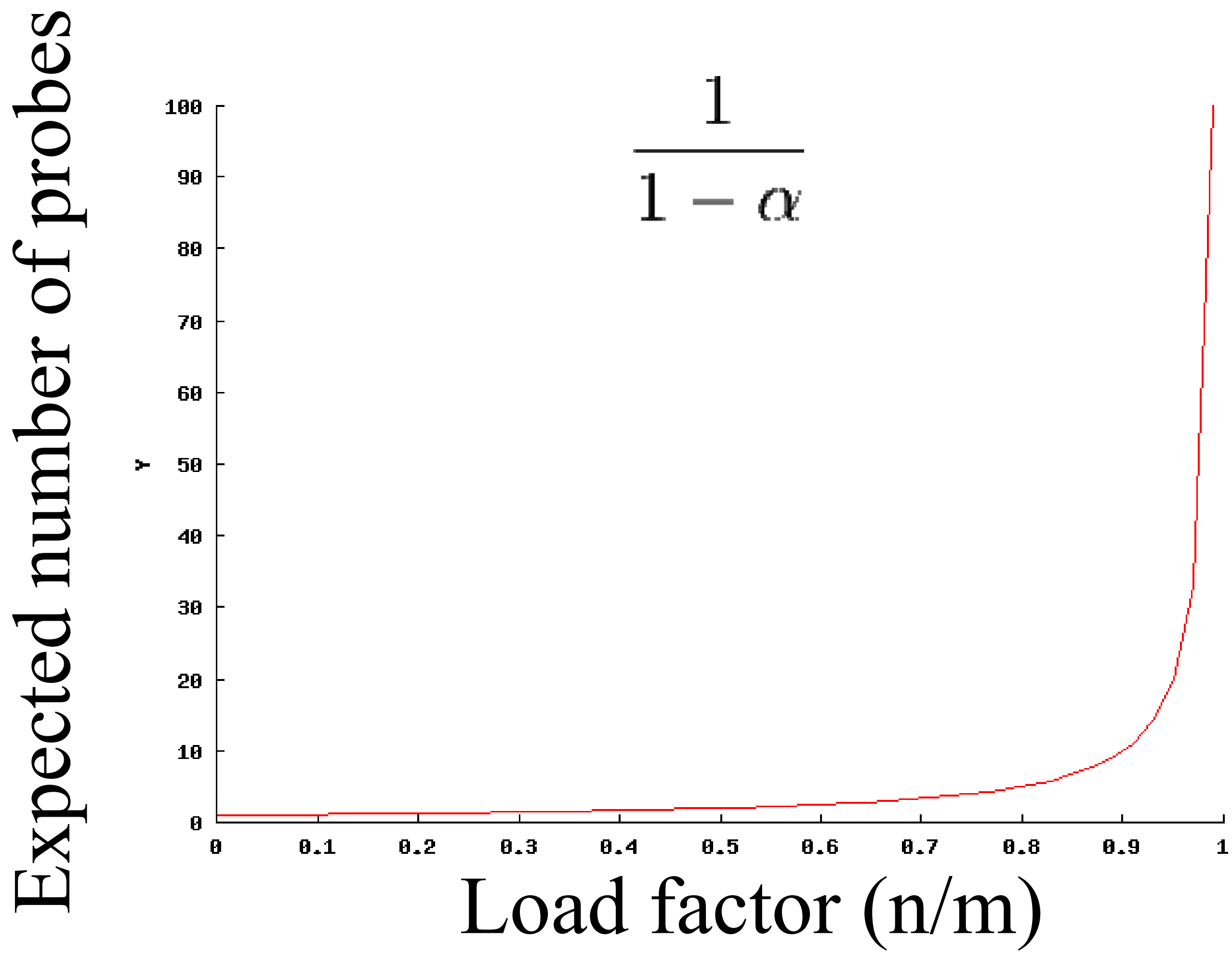
e.g.:
$\alpha = n/m = 90\%$
$p = 0.1$

➔ every trial succeeds with probability ≥p

expected number of probes till success? $\leq \dfrac{1}{p} = \dfrac{1}{1-\alpha}$

e.g. if α=90%, expected number of probes is at most 10

Expected number of probes



Open addressing sensitive to α

As $\alpha \rightarrow 1$, access time shoots up

Open Addressing vs. Chaining

- Open addressing skips linked lists
  - Saves space (of list pointers)
  - Better locality of reference
    - Array concentrated in m space
    - So fewer main-memory accesses bring it to cache
    - Linked list can wander all of memory
- Open addressing sensitive to α
  - As $\alpha \rightarrow 1$, access time shoots up
  - Cannot allow $\alpha > 1$
- Open addressing needs good hash to avoid clustering

Today: Hashing III: Space issues

Rev: Hash functs, chaining, SUHA, rolling, signatures

- ✓ **Dynamic dictionaries:** Resizing hash tables
  - ✓ When to resize: insertions, deletions
  - ✓ Resizing operations, amortized analysis
- ✓ **Open addressing:** Doing away w/ linked lists
  - ✓ Operations: insertion, deletion
  - ✓ Probing: linear probing, double hashing
  - ✓ Performance analysis: UHA, open vs. chaining
- ➢ **Advanced topics:** Randomized algorithms (shht!)
  - ❑ Universal hashing, perfect hashing
  - ❑ Fingerprinting, file signatures, false positives

Done with unit #2: Hashing

- **Last Tues: Genomes, Dictionaries, Hashing**
  - Intro, basic operations, collisions and chaining
  - Simple uniform hashing assumption
- **Last Thur: Faster hashing, hash functions**
  - Hash functions in practice: div/mult/python
  - Faster hashing: Rolling Hash: $O(n^2) \rightarrow O(n \lg n)$
  - Faster comparison: Signatures: mismatch time
- **Today: Space issues**
  - Dynamic resizing and amortized analysis
  - Open addressing, deletions, and probing
  - Advanced topics: universal hashing, fingerprints

Next week: Sorting

Unit	Pset	Week	Date		Lecture (Tuesdays and Thursdays)		Recitation (Wed and Fri)
Intro	PS1	1	Tue Feb 01	1	Introduction and Document Distance	1	Python and Asymptotic Complexity
Binary	Out: 2/1		Thu Feb 03	2	Peak Finding Problem	2	Peak Finding correctness & analysis
Search	Due: Mon 2/14	2	Tue Feb 08	3	Scheduling and Binary Search Trees	3	Binary Search Tree Operations
Trees	HW lab: Sun 2/13		Thu Feb 10	4	Balanced Binary Search Trees	4	Rotations and AVL tree deletions
Hashing	PS2 Out: 2/15	3	Tue Feb 15	5	Hashing I : Chaining, Hash Functions	5	Hash recipes, collisions, Python dicts
	Due: Mon 2/28		Thu Feb 17	6	Hashing II : Table Doubling, Rolling Hash	6	Probability review, Pattern matching
	HW lab:Sun 2/27	4	Tue Feb 22	-	President's Day - Monday Schedule - No Class	-	No recitation
			Thu Feb 24	7	Hashing III : Open Addressing	7	Universal Hashing, Perfect Hashing
Sorting	PS3. Out: 3/1	5	Tue Mar 01	8	Sorting I : Insertion & Merge Sort, Master Theorem	8	Proof of Master Theorem, Examples
	Due: Mon 3/7		Thu Mar 03	9	Sorting II : Heaps	9	Heap Operations
	HW lab: Sun 3/6	6	Tue Mar 08	10	Sorting III: Lower Bounds, Counting Sort, Radix Sort	10	Models of computation
			Wed Mar 09	Q1	Quiz 1 in class at 7:30pm. Covers L1-R10. Review Session on Tue 3/8 at 7:30pm.		
Graphs	PS4. Out: 3/10		Thu Mar 10	11	Searching I: Graph Representation, Depth-1st Search	11	Strongly connected components
and	Due: Fri 3/18	7	Tue Mar 15	12	Searching II: Breadth-1st Search, Topological Sort	12	Rubik's Cube Solving
Search	HW lab:W 3/16		Thu Mar 17	13	Searching III: Games, Network properties, Motifs	13	Subgraph isomorphism
Shortest	PS5	8	Tue Mar 29	14	Shortest Paths I: Introduction, Bellman-Ford	14	Relaxation algorithms
Paths	Out: 3/29		Thu Mar 31	15	Shortest Paths II: Bellman-Ford, DAGs	15	Shortest Path applications
	Due: Mon 4/11	9	Tue Apr 05	16	Shortest Paths III: Dijkstra	16	Speeding up Dijkstra's algorithm
	HW lab:Sun 4/10		Thu Apr 07	17	Graph applications, Genome Assembly	17	Euler Tours
Dynamic	PS6	10	Tue Apr 12	18	DP I: Memoization, Fibonacci, Crazy Eights	18	Limits of dynamic programming
Program	Out: Tue 4/12		Wed Apr 13	Q2	Quiz 2 in class at 7:30pm. Covers L11-R17. Review Session on Tue 4/13 at 7:30pm.		
ming	Due: Fri 4/29		Thu Apr 14	19	DP II: Shortest Paths, Genome sequence alignment	19	Edit Distance, LCS, cost functions
	HW lab:W 4/27	11	Tue Apr 19	-	Patriot's Day - Monday and Tuesday Off	-	No recitation
			Thu Apr 21	20	DP III: Text Justification, Knapsack	20	Saving Princess Peach
		12	Tue Apr 26	21	DP IV: Piano Fingering, Vertex Cover, Structured DP	21	Phylogeny
Numbers	PS7 out Thu4/28		Thu Apr 28	22	Numerics I - Computing on large numbers	22	Models of computation return!
Pictures	Due: Fri 5/6	13	Tue May 3	23	Numerics II - Iterative algorithms, Newton's method	23	Computing the nth digit of π
(NP)	HW lab: Wed 5/4		Thu May 5	24	Geometry: Line sweep, Convex Hull	24	Closest pair
		14	Tue May 10	25	Complexity classes, and reductions	25	Undecidability of Life
Beyond			Thu May 12	26	Research Directions (15 mins each) + related classes		
		15	Finals week	Q3	Final exam is cumulative L1-L26. Emphasis on L18-L26. Review Session on Fri 5/13 at 3pm		