10/31/18 415

Project 3 is up
- Due Tuesday 11/13
Compare efficiency 2-3 v. BST

Main goal to implement 2-3 tree

---

probability 2 people $\frac{1}{365}$ $\frac{364}{365}$

3 people

$\frac{365}{365}$ $\frac{364}{365}$

$\frac{n!}{n} = (n-1)!$

$\frac{n!}{365^n}$

no two people

$\frac{365!}{(365-n)!\,365^n}$

at least two people $1 - \left(\frac{365!}{(365-n)!\,365^n}\right)$

Collision Types

Open Hashing

each cell is header of linked list

$$h(C_0 \ldots C_n) = \left(\sum_{0}^{n} pv(C_i)\right) \% 13$$

if evenly distributed

length of list = load factor = $\alpha = n/m$

Average probes in S(uccessful) or U(nsuccessful)

$S \approx 1 + \alpha/2 \quad U = \alpha$

1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	C	D	E	F	G	H	I	J	K	L	M

14	15	16	17	18	19	20	21	22	23	24	25	26
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

$$13 \overline{)48} \quad 3 \text{ r } 9$$

$$13 \overline{)24} \quad 1 \text{ r } 11 \quad \boxed{11} = h(\text{key})$$

$$\begin{array}{r} 30 \\ \cdot 2 \\ \hline 6 \\ \hline 48 \end{array}$$

Load α is typically kept small (ideally close to 1)

Closed Hashing:

each slot used 1x

linear probing → probe til next open slot

$$S \approx \frac{1}{2}\left(1 + \frac{1}{1-\alpha}\right) \text{ and } U \approx \frac{1}{2}\left(1 + \frac{1}{(1-\alpha)^2}\right)$$

11/5 40'

Announcements

Project 3 Due next week

HW 5 TBD

Hashing

- A very efficient way of implementing a dictionary with:
  - insert
  - delete
  - find

How to implement

closed: don't

→ if

Hashing Application

- symbol tables (ie) table of a compiler's generation of program's symbols)
- AI applications store predictions and probabilities of
- Databases (extendible hashing)

Open
~~Closed~~ $S(\alpha) = 1 + \frac{1}{1-\alpha}$

Dynamic Programming

Main idea
set up recurrence relationships and compute and store subproblems into a table instead of recomputing them

Coin-row problem

DP solution
$F(n)$ maximum value of non adjacent coins last coin is C_i

divide into two groups

$F(n) = C_i$	$F(n) = C_{i-1} + F(n-1)$
$F(n) = C_i + F(n-2)$	$F(n) = C_{i-1} + F(n-3)$

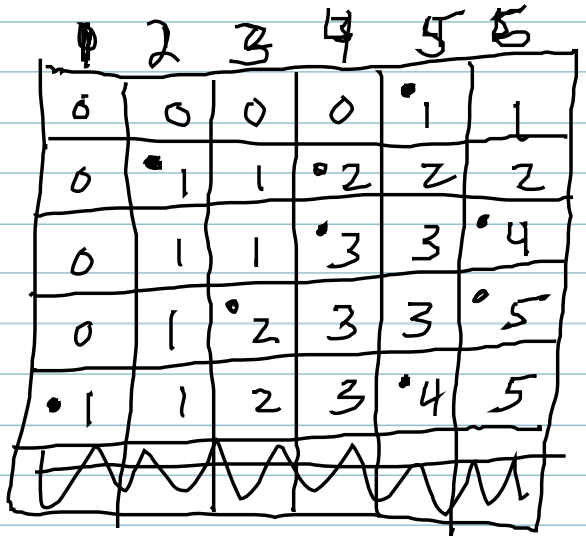
Rec Rel

$$F(n) = \max\{C_n + F(n-2), F(n-1)\} \quad n > 1$$

index	0	1	2	3	4	5	6
coins	-	5	1	2	10	6	2
F( )	0	5					

max(

Ex 3 Coin-collecting by robot



$F(i,j) = \max(F(i-1,j), F(i,j-1)) + C_{ij}$

Rec Rel $F(i,j) = \max(F(i-1,j), F(i,j-1)) + c_{ij}$ for $1 \le i \le n$, $1 \le j \le m$

, $c_{ij} = 1$

11/19 4:15

Greedy

Prim's — network

Greedy strategy = Optimal solutions
- Huffman Codes : simple scheduling
- normal coin denominations -
- MST
- single source shortest path

Approximations
- TSP
- knapsack
- other combinatory

Dijkstra's need a source

Prim's

Goal = not total from source. Is shortest path (Minimal Spanning Tree)

✓	✓	X	X	✓	✓	✓	X	X	✓
1	2	3	4	5	6	7	8	9	10
E	C	C	C	F	F	B	T	C	A

11/

Announcements

Project 4 due Sun 12/2

3 more lectures!

HW6 will be up 11/28
- due 12/5

CS 330 Game Programming
- ~~alot of~~ quite a bit vector math
- if haven't taken linear algebra still ok
  - will give tools for vector math needed

Plazza question

All games:
- engine
  - input
  - collision physics
  - AI

Game Play
Games. what are rules?
what are characters?

Labs will focus on gameplay using unReal game engine

Lecture focus on both
midterm → 3 Projects → weekly labs, one lab VR
no final → Final Project instead, will make game

1st Two projects individual
Final project 3 or 4

Greedy Technique

Optimal Solutions

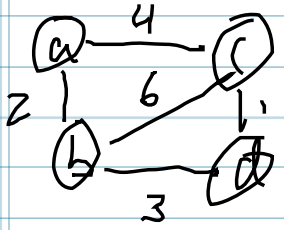
- Project
  Creat bottom up max heap

  show greedy speed at cost of optimality

P.II Extra Credit involves approach setting $O(n \cdot k \log n)$

PI Build path on go instead of backtracing

Prim's MST algorithm
use for solving TSP



start w/ any

Start w/ a

Tree				
a(-,0)	c(a,4)	b(a,2)	d(-,∞)	a
b(a,2)	c(a,4)	d(b,3)		b
				d

d(-,0)	c(d,1), b(d,3), a(-,∞)	d → c → d → b → c → a
c(d,1)	b(d,3) a(c,4)	
b(d,3)	a(b,2)	

same complexity as Dijkstra's

$|V| * ($Find min + update distances$)$

unordered array or min Heap

How many

weighted adjacency matrix → $|V|^2$

adjacency list → $|E| \log |V|$

same as dijkstra's

Complete graph $|E| \approx |V|^2$

1st option better for complete graph
2nd much better for sparse graph

Practice Drawing lines on graph from slide show

- Shading inequalities B4 next class