Lecture 19: Monday February 24, 2003

today

- dynamic programming [CLRS Ch. 15.1-3]
  - chained matrix multiplication

soon

- union find [CLRS Ch. 21]
- graph algorithms [CLRS Ch. 22,23]

announcements

(recall) 2nd approach to chain matrix mult'n: recursion

- let $M(x, y)$ be the minimum number of scalar multiplications needed to perform $A_x \times A_{x+1} \times \ldots \times A_y$
- $M(x,y) = \begin{cases} 0 & \text{if } x = y \\ \min_{x \le t < y}\{M(x,t) + M(t+1,y) + d_{x-1}d_t d_y\} & \text{if } x < y \end{cases}$
- e.g. $M(1,4) = \min \left\{ \begin{array}{l} M(1,1) + M(2,4) + d_0 d_1 d_4, \\ M(1,2) + M(3,4) + d_0 d_2 d_4, \\ M(1,3) + M(4,4) + d_0 d_3 d_4 \end{array} \right\}$
- implementation: recursion

```
proc M(x,y)
  if x=y then return 0
  else
    cost <- infty
    for t <- x to y-1 do
      new <- M(x,t)+ M(t+1,y)+ d[x-1]d[t]d[y]
      if new < cost then
        cost <- new
    return cost
```

```
                          14

11        24          12    34          13        44

   22 34     23 44 11 22   33 44 11 23     12 33

     33 44 22 33                  22 33 11 22
```

- let T_n be total number of $M(\,)$ calls made by $M(x, x+n-1)$
- $T_n = \begin{cases} 1 & \text{if } n = 1 \\ 1 + \sum_{j=1}^{n-1}(T_j + T_{n-j}) = 1 + 2\sum_{j=1}^{n-1} T_j & \text{if } n \geq 2 \end{cases}$
- show by induction: $T_n = 3^{n-1}$ for $n \geq 1$
- run time for $M(1,n) \in \Omega(3^n)$ ☹

- 3rd approach: memoization

```
exercise
```

- 4th approach: full dynamic programming (with iteration)

```
proc dpM(n)
  for j <- 1 to n do
    M[j,j]<-0
  for shift <- 1 to n do
    for x <- 1 to n-shift do
      y <- x+shift
      cost <- infty
      for t <- x to y-1 do
        new <- M[x,t]+ M[t+1,y]+ d[x-1]d[t]d[y]
        if new < cost then
          cost <- new
      M[x,y] <- cost
  return M[1,n]
```

- trace: original example

0			
	0		
		0	
			0

*			
	*		
		*	
			*

$$M[1,2] \leftarrow \min\{M[1,1] + M[2,2] + 5 \cdot 2 \cdot 6\} \leftarrow$$

$$M[2,3] \leftarrow \min\{M[2,2] + M[3,3] + 2 \cdot 6 \cdot 4\} \leftarrow$$

$$M[3,4] \leftarrow \min\{M[3,3] + M[4,4] + 6 \cdot 4 \cdot 3\} \leftarrow$$

$$M[1,3] \leftarrow \min\left\{ \begin{array}{l} M[1,1] + M[2,3] + 5 \cdot 2 \cdot 4, \\ M[1,2] + M[3,3] + 5 \cdot 6 \cdot 4 \end{array} \right\} \leftarrow$$

$$M[2,4] \leftarrow \min\left\{ \begin{array}{l} M[2,2] + M[3,4] + 2 \cdot 6 \cdot 3, \\ M[2,3] + M[4,4] + 2 \cdot 4 \cdot 3 \end{array} \right\} \leftarrow$$

$$M[1,4] \leftarrow \min\left\{ \begin{array}{l} M[1,1] + M[2,4] + 5 \cdot 2 \cdot 3, \\ M[1,2] + M[3,4] + 5 \cdot 6 \cdot 3, \\ M[1,3] + M[4,4] + 5 \cdot 4 \cdot 3 \end{array} \right\} \leftarrow$$

- time in $\Theta(n^3)$ 

other problems suited to dynamic programming

- constructing optimal binary search trees
- string matching: longest common subsequence
- all pairs shortest paths in (di)graph