For Friday April 27 @11pm: Please submit a self evaluation for HW 9
Basic Algorithms HW10: Due May 1@ 11pm

11.anotherDP. Let $A[1..n]$ be an array of n positive and negative numbers. We wish to partition the sequence of n values into k subsequences of consecutive values

$$A[1], A[2], \ldots, A[i_1 - 1] \;||\; A[i_1], A[i_1 + 1], \ldots, A[i_2] - 1 \;||\; \cdots \;||\; A[i_{k-1}], A[i_{k-1} + 1], \ldots, A[n],$$

where the sum of the square of the sum of numbers in each subsequence is as small as possible, and there is no constraint on the value k: it can be any count in $1..n$.

Example: $A[1..n]$ contains $3, -4, 19, -16, 1, 2, -4, 3, 2$. Then the partitioning $3, -4, 19, -16 \;||\; 1 \;||\; 2 \;||\; -4, 3, 2$ gives the best possible result: $(3 - 4 + 19 - 16)^2 + 1^2 + 2^2 + (-4 + 3 + 2)^2 = 10$. Yes, there is another partitioning that is equally good.

Present a high-level recursive program specification for the solution.

11.50

5.Graph. Let $G = V, \Xi$ be a directed graph where $V = \{1, 2, 3, \ldots, n\}$, and the edges E are presented as a set (or array $Edges[1..|\Xi|]$ of ordered pairs. Cive a high-level $\Theta(n + |E|)$-time algorithm to build the array of adjacency lists $Adj[1..n]$ for G with each list $Adj[i]$ storing the neighbors of i in sorted order. Hint: the time bound dictates the method.

Hint:

Did you say something about the second pass of the Paige-Tarjan lexicographical sorting algorithm?

From Chapter 5: 5.40a

From Chapter 6: 6.5

6insertdelete.a Show the 23tree that results from inserting the numbers in the order listed: 1,3,4,9,8,7,2,6. The leaf-level groupings from left to right should comprise: 3 siblings followed by 3 siblings followed by 2 siblings.

6insertdelete.b. Now show what happens when 5 is inserted.

6insertdelete.c. Now show what happens when 5 is deleted. Note that the depth did not change.

6.10a Hint: you need the following language: On the path from the root of the 23tree down to the vertex that would become the parent of the leaf-records x (if no splitting were to occur), each . . . .

6.range. Briefly explain how to implement standard range queries with 23trees:
For a tree with $n > 1$ records, briefly explain how to implement a structure where all of the following operations run in $O(\log n)$ time, for $n > 1$.

a $Insert(x)$. Assume that insertions are only executed for a key that is not yet in the tree.

b $Locate(x)$, which returns a pointer to the record with key x, and `Nil` if there is no such record.

c $Delete(x)$.

d $CountLessThan(x)$, which returns the number of elements less than x.

6.intervals. Explain how to use two 23Trees to store intervals. An interval $[a, b)$ is defined by the pair of real values $a < b$, and denotes the collection of points x on the x-axis where $a \leq x < b$. Note that there is no merging of intervals: if you insert $[0, 3)$ and then insert $[3, 5)$, the structure will store those two intervals as separate records, and never merge them into form $[0, 5)$.
Explain why the implementation can perform each of the following operations in $O(\log n)$ time for $n > 1$. The operations are:

a $Insert(a, b)$, which denotes the interval $[a, b)$. Assume that all intervals in T are unique.

b $Locate(a, b)$.

c $Delete(a, b)$.

6.intervalstabs. Explain how to use two 23Trees to store intervals as in 6.intervals above, and to compute $Stab(x)$ in O(log time for $n > 1$.

$Stab(x)$ is the number of intervals $[a, b)$ in the tree where $a \leq x < b$, so that the stabbing number is the number of intervals that are in the tree and that contain the point x.

Hint: According to the waybak time machine, there have been exactly 10,970,812,004 humans have been born since the beginning of time, and 4,130,305,001 humans have died since the beginning of time. Can you deduce the number of humans who are alive? Now think of the x-axis as time. $Stab(x)$ is the number of intervals that have been born by time x, but have not yet died.

8.qqq

Let G be a graph with vertices $\{1, 2, 3, \ldots, n\}$ and let R, W, and B be the graph's $n \times n$ edgecost arrays for G's red, white, and blue edges respectively. You can (if necessary) also use the $n \times n$ array Z, which satisfies $Z[i, i] = 0$ for all i, and $Z[i, j] = \infty$ for all $i \neq j$.

a) As always, $R[i, j]$ is infinite if there is no red edge from i to j, and the same is true for W and B. Solve the APSP problem for paths with the edge color pattern in $R^+W^+B^+$, where R^+ means a path of one or more red edges, so that $R^+W^+B^+$ restricts paths to begin with one or more red edges, and then continue with one or more white edges, and finish with one or more blue edges. Present the edgecost array for the supergraph, and explain how to access the supersolution array to determine Pcost[i; j].

b) Suppose that the F-W algorithm uses cn3 +O(n2) operations. What is the corresponding operation count for your solution to par a)?

c) We did not cover the following methof in class, but it is in the book. You will not see a problem like this on the final. We briefly covered the design of supergraphs to solve constrained shortest path problems where the path constraint was that each solution path had colored edges that matched some specialized color pattern. The decomposition method builds superedges with color patterns. For example, the following 2-step computation builds super-edges that are really a red edge followed by a blue edge:

$$RBcost[i, j] = \min_k \{Rcost[i, k] + Bcost[k, j]\}.$$

By repeatedly applying two-step builders and FW solvers on $n \times n$ edgecost arrays, you can solve anything that can be solves by supergraphs, and the efficiency is better.

So you are welcome to think about or solve the following.

Suppose that you have lost your crayons, so that you cannot use programming by picture. And suppose that due to some kind of programming error, you can only run the F-W algorithm on $n \times n$ edge cost arrays, and not on $3n \times 3n$ arrays. Luckily, you also have the following Best2Step algorithm:

```
procedure Best2Step(n,EcostA[1..n,1..n],EcostB[1..n,1..n];;ABcost[1..n,1..n]);
  output: ABcost[i,j] = least cost of a path from i to j built from an A-edge followed by a B-edge;
  Set all entries in ABcost to ∞;
  Set all EcostA[i,i] and EcostB[i,i] entries to ∞;
  for i ← 1 to n do
    for j ← 1 to n do
      for k ← 1 to n do
        ABcost[i,j] ← min{ABcost[i,j], Acost[i,k] + Bcost[k,j]}
      endfor
    endfor
  endfor
end_Best2Step;
```

This code will also only work on arrays of size $n \times n$.

The problem: Explain how to solve part a with these utilities.

d) Assume that $Best2Step$ has the same operation count as the F-W algorithm. What is the operation count for your solution

in part c?

11.14. which is the most elegant divide-and-conquer puzzle ever.

8.onemore Interpreting supergraph edge cost arrays Let R be an $n \times n$ edge cost array for red edges, and let W and B be the same for the colors white and blue. For specificity, let the diagonal entries of R, W, and B be infinite. Define the $3n \times 3n$ superarray S to have the following structure with the first row and first column located at the $*$s in the upper left corner:

$$S = \begin{pmatrix} * & * & * \\ R & W & B \\ \infty & W & B \\ \infty & \infty & B \end{pmatrix},$$

a) Suppose we run the Flyod-Warshall algorithm on S to get the resulting $3n \times 3n$ path cost array P. Define the $n \times n$ array Q by $Q[i,j] = P[i, j+2n]$, so that $Q[i,j]$ gives the shortest path from i to j where the edges in the path have costs defined by R, W and B, and the paths have edge colors that form some kind of color pattern that can be defined by a regular expression in r, w, and b. What is that expression?

b) Suppose that S and P are computed as in part a, and we define the $n \times n$ array R so that $R[i,j] = P[i,j]$. What shortest path problem does R solve?

c) Suppose that S and P are computed as in part a, and we define the $n \times n$ array T so that $T[i,j] = P[i, j+n]$. What shortest path problem does T solve?

8.last Interpreting supergraph edge cost arrays Let G R be an $n \times n$ edge cost array for red edges, and let W and B be the same for the colors white and blue. For specificity, let the diagonal entries of R, W, and B be infinite. Define the $3n \times 3n$ superarray S to have the block structure

$$S = \begin{pmatrix} * & * & * \\ \infty & R & \infty \\ \infty & \infty & W \\ \infty & \infty & B \end{pmatrix},$$

where the first row and first column of the array are located at the $*$s in the upper left of the structure.

a) Suppose we run the Flyod-Warshall algorithm on S to get the resulting $3n \times 3n$ path cost array P. Define the $n \times n$ array Q by $Q[i,j] = P[i, j+2n]$, so that $Q[i,j]$ gives the shortest path from i to j where the edges in the path have costs defined by R, W and B, and the paths have edge colors that form some kind of color pattern that can be defined by a regular expression in r, w, and b. What is that expression?

b) Suppose we define the $3n \times 3n$ superarray $T = \begin{pmatrix} * & * & * \\ R & R & R \\ \infty & W & B \\ \infty & \infty & B \end{pmatrix}$, where the first row and first column of the array are located at the $*$s in the upper left of the structure. Suppose we run the Flyod-Warshall algorithm on T to get the resulting $3n \times 3n$ path cost array P. Define the $n \times n$ array Q by $Q[i,j] = P[i, j+2n]$. What shortest path problem does Q solve?

c) Suppose we define the $3n \times 3n$ superarray $U = \begin{pmatrix} * & * & * \\ R & W & B \\ \infty & W & B \\ Z & \infty & B \end{pmatrix}$, where the first row and first column of the array are located at the $*$s in the upper left of the structure, and $Z[i,j] = \begin{cases} 0, & \text{if } i = j; \\ \infty, & \text{if } i \neq j. \end{cases}$ Suppose we run the Flyod-Warshall algorithm on U to get the resulting $3n \times 3n$ path cost array P. Define the $n \times n$ array Q by $Q[i,j] = P[i, j+2n]$. What shortest path problem does Q solve?