CS61B Lecture #36

gramming

 Trip: Enumeration types.

Dynamic Programming

Garcia):

n a list with an even number of non-negative integers.

er in turn takes either the leftmost number or the

.

 get the largest possible sum.

rting with (6, 12, 0, 8), you (as first player) should take
ever the second player takes, you also get the 12, for a

r opponent plays perfectly (i.e., to get as much as pos-
an you maximize your sum?

s with exhaustive game-tree search.

Obvious Program

kes it easy, again:

```
n(int[] V) {
l, i, N = V.length;
 0, total = 0; i < N; i += 1) total += V[i];
estSum(V, 0, N-1, total);


rgest sum obtainable by the first player in the choosing
n the list V[LEFT .. RIGHT], assuming that TOTAL is the
 all the elements in V[LEFT .. RIGHT].  */
n(int[] V, int left, int right, int total) {
 > right)
 0;

= total - bestSum(V, left+1, right, total-V[left]);
= total - bestSum(V, left, right-1, total-V[right]);
 Math.max(L, R);
```

$C(0) = 1,\ C(N) = 2C(N-1)$; so $C(N) \in \Theta(2^N)$

Still Another Idea from CS61A

 is that we are recomputing intermediate results many

emoize the intermediate results. Here, we pass in an
 ($N = $ V.length) of memoized results, initialized to -1.

```
n(int[] V, int left, int right, int total, int[][] memo) {
 > right)
 0;
(memo[left][right] == -1) {
= total - bestSum(V, left+1, right, total-V[left], memo);
= total - bestSum(V, left, right-1, total-V[right], memo);
eft][right] = Math.max(L, R);

emo[left][right];
```

nber of recursive calls to bestSum must be $O(N^2)$, for
gth of V, an enormous improvement from $\Theta(2^N)$!

Iterative Version

 recursive version, but the usual presentation of this
as *dynamic programming*—is iterative:

```
n(int[] V) {
memo = new int[V.length][V.length];
total = new int[V.length][V.length];
 i = 0; i < V.length; i += 1)
][i] = total[i][i] = V[i];
 k = 1; k < V.length; k += 1)
nt i = 0; i < V.length-k-1; i += 1) {
l[i][i+k] = V[i] + total[i+1][i+k];
L = total[i][i+k] - memo[i+1][i+k];
R = total[i][i+k] - memo[i][i+k-1];
[i][i+k] = Math.max(L, R);

emo[0][V.length-1];
```

figure out ahead of time the order in which the memo-
will fill in memo, and write an explicit loop.

e needed to check whether result exists.

y bother unless it's necessary to save space?

Longest Common Subsequence

d length of the longest string that is a subsequence of
other strings.

ngest common subsequence of
lls␣sea␣shells␣by␣the␣seashore" and
ld␣salt␣sellers␣at␣the␣salt␣mines"

␣sells␣␣the␣sae" (length 23)

sting, for example.

rsive algorithm:

```
 of longest common subsequence of S0[0..k0-1]
[0..k1-1] (pseudo Java) */
 lls(String S0, int k0, String S1, int k1) {
= 0 || k1 == 0) return 0;
0-1] == S1[k1-1]) return 1 + lls(S0, k0-1, S1, k1-1);
urn Math.max(lls(S0, k0-1, S1, k1), lls(S0, k0, S1, k1-1);
```

 but obviously memoizable.

oized Longest Common Subsequence

```
ngest common subsequence of S0[0..k0-1]
-1] (pseudo Java) */
tring S0, int k0, String S1, int k1) {
 new int[k0+1][k1+1];
 : memo) Arrays.fill(row, -1);
 k0, S1, k1, memo);


nt lls(String S0, int k0, String S1, int k1, int[][] memo) {
 k1 == 0) return 0;
1] == -1) {
 == S1[k1-1])
1] = 1 + lls(S0, k0-1, S1, k1-1, memo);

1] = Math.max(lls(S0, k0-1, S1, k1, memo),
              lls(S0, k0, S1, k1-1, memo));

][k1];
```

ill the memoized version be?

oized Longest Common Subsequence

```
ngest common subsequence of S0[0..k0-1]
-1] (pseudo Java) */
tring S0, int k0, String S1, int k1) {
 new int[k0+1][k1+1];
 : memo) Arrays.fill(row, -1);
 k0, S1, k1, memo);


nt lls(String S0, int k0, String S1, int k1, int[][] memo) {
 k1 == 0) return 0;
1] == -1) {
 == S1[k1-1])
1] = 1 + lls(S0, k0-1, S1, k1-1, memo);

1] = Math.max(lls(S0, k0-1, S1, k1, memo),
              lls(S0, k0, S1, k1-1, memo));

][k1];
```

ill the memoized version be? $\Theta(k_0 \cdot k_1)$

Trip into Java: Enumeration Types

ed a type to represent something that has a few, named,
ies.

st form, the only necessary operations are == and !=;
perty of a value of the type is that it differs from all

sions of Java, used named integer constants:

```
ces {
IECE = 0,    // Fields in interfaces are static final.
ING = 1,
IECE = 2,
ING = 3,
 4;
```

ovide *enumeration types* as a shorthand, with syntax like

```
BLACK_PIECE, BLACK_KING, WHITE_PIECE, WHITE_KING, EMPTY };
```

these values are basically **ints**, accidents can happen.

Enum Types in Java

of Java allows syntax like that of C or C++, but with
tees:

```
iece {
, BLACK_KING, WHITE_PIECE, WHITE_KING, EMPTY
```

e as a new reference type, a special kind of class type.

LACK_PIECE, etc., are static, final *enumeration constants*
s) of type PIECE.

tomatically initialized, and are the only values of the
type that exist (illegal to use **new** to create an enum

e ==, and also switch statements:

```
g(Piece p) {
{
K_KING: case WHITE_KING: return true;
return false;
```

king Enumerals Available Elsewhere

e BLACK_PIECE are static members of a class, not classes.

nlike C or C++, their declarations are not automatically
le the enumeration class definition.

classes, must write Piece.BLACK_PIECE, which can get

th version 1.5, Java has *static imports:* to import all
tions of class checkers.Piece (including enumerals), you

```
ic checkers.Piece.*;
```

port clauses.

use this for enum classes in the anonymous package.

Operations on Enum Types

claration of enumeration constants significant: .ordinal()
sition (numbering from 0) of an enumeration value. Thus,
_KING.ordinal() is 1.

ece.values() gives all the possible values of the type.
n write:

```
: Piece.values())
t.printf("Piece value #%d is %s%n", p.ordinal(), p);
```

unction Piece.valueOf converts a String into a value of
So Piece.valueOf("EMPTY") == EMPTY.

Fancy Enum Types

asses. You can define all the extra fields, methods, and
 you want.

 are used only in creating enumeration constants. The
arguments follow the constant name:

```
(BLACK, false, "b"), BLACK_KING(BLACK, true, "B"),
(WHITE, false, "w"), WHITE_KING(WHITE, true, "W"),
 false, " ");

al Side color;
al boolean isKing;
al String textName;

color, boolean isKing, String textName) {
r = color; this.isKing = isKing; this.textName = textName;


) { return color; }
ing() { return isKing; }
Name() { return textName; }
```