Tell It What to Know

6.871 - Lecture 2

A Reminder

- Checkbook balancing vs.
  getting out of the supermarket
- Character of task
- Character of solution
- Go past image to technical ideas and concepts

Purposes of This Lecture

- Explain the mindset of knowledge engineering
- Change your mind about what a program is
  - From a buncha bits to …
  - From code to …
- Change your mind about how to create them
  - Don't tell it what to do
  - Build it incrementally
- Change your mind about what to use a computer for
  - Many things…

Punchlines

- The issue is style and pragmatics, not theory
- A program can be much more than just code.
  It can be a repository of knowledge,
  an environment for the development of knowledge
- Embody the reasoning, not (just) the calculation
- Don't tell it what to do, tell it what to know, and how to use what it knows (often many different ways)
  - Task changes from writing a program to specifying the knowledge.
  - Task becomes debugging knowledge, not code.

Punchlines

- One payoff: multiple uses of the same knowledge.
- Performance is only the beginning
  Solving the problem is only (a small) part of the job
  - Explanation
  - Learning
  - Tutoring
- Suppressing detail helps
- Build a custom language

Punchlines

- Nothing is ever right the first time
  - Nature of the task
  - Nature of the knowledge
  - Evolutionary development
    - Build a little
    - Test a little
    - Redesign a little

What's a Good Representation?

- Consider: 1996 vs. MCMXCVI

- Which would you rather use in arithmetic? Why?
  - Makes important things obvious
  - Syntax and semantics are simple, consistent
  - Algorithms for use are simple

What's a Good Representation?

- Consider: 1996 vs. 11111001100

- Which would the computer rather use in arithmetic? Why?
  - Algorithms for use are simple
  - And simplicity is in the eye of the interpreter

The Power of A Good Representation

The proportional ownership of the first party shall be equal to a ratio, the numerator of which is: a ratio, the numerator of which is the holding period of the first party multiplied by the capital contributed by the first party, and the denominator of which is a sum, the first term of which is the holding period of the first party and the second term of which is the holding period of the second party; and a denominator which is the sum of two terms; the first term of which is a ratio, the numerator of which is the holding period of the first party multiplied by the capital contributed by the first party, and the denominator of which is a sum, the first term of which is the holding period of the first party, the second term of which is the holding period of the second party; and the second term of which is a ratio, the numerator of which is the holding period of the second party multiplied by the capital contributed by the second party, and the denominator of which is a sum, the first term of which is the holding period of the first party and the second term of which is the holding period of the second party.

The proportional ownership of the first party shall be equal to a ratio, the numerator of which is: a ratio, the numerator of which is the holding period of the first party multiplied by the capital contributed by the first party, and the denominator of which is a sum, the first term of which is the holding period of the first party and the second term of which is the holding period of the second party; and a denominator which is the sum of two terms; the first term of which is a ratio, the numerator of which is the holding period of the first party multiplied by the capital contributed by the first party, and the denominator of which is a sum, the first term of which is the holding period of the first party, the second term of which is the holding period of the second party; and the second term of which is a ratio, the numerator of which is the holding period of the second party multiplied by the capital contributed by the second party, and the denominator of which is a sum, the first term of which is the holding period of the first party and the second term of which is the holding period of the second party.

$$\frac{\frac{t1 * m1}{t1 + t2}}{\frac{t1 * m1}{t1 + t2} + \frac{t2 * m2}{t1 + t2}}$$

What's a Program?
The Minimal Number of Bits View

```
      DO 14 I = 1,N
      DO 14 J = 1,N
14    V(I,J) = (I/J)*(J/I)
```

What's a Program?
The Minimal Number of Bits View

```
      DO 14 I = 1,N
      DO 14 J = 1,N
14    V(I,J) = (I/J)*(J/I)
```

```
#include <stdio.h>
main ( )
{int v[5][5];
 int i,j;
 for (i=1; i<5; i++)
  for (j=1; j<5; j++)
   v[i][j]=(i/j)*(j/i)}
```

Task: Symbolic Mathematics

How can we take a derivative of

$$3x^3 + 4x^2 + 5x + 7$$

to get

$$9x^2 + 8x + 5$$

Version 1

```
PROCEDURE READPROBLEM (REAL ARRAY P)
  Read in one line of integers, the
  coefficients of a polynomial, into array P.
  Also sets DEGREE to degree of polynomial.
  Example:
```

$$3x^3 + 4x^2 + 5x + 7$$

```
  is entered by typing
                        3 4 5 7

 PROCEDURE POLY-DIFF (REAL ARRAY PROBLEM)
      FOR I = DEGREE TO 1 STEP -1 DO
           ANSWER [I-1] = I * PROBLEM [I]
```

Version 2

```
PROCEDURE POLY-DIFF (REAL ARRAY PROBLEM)
FOR I = DEGREE TO 1 STEP -1 DO
 ANSWER [COEFF, I] = PROBLEM [EXPON, I] *
                     PROBLEM [COEFF, I]
 ANSWER [EXPON, I] = PROBLEM [EXPON, I] - 1
```

But What About:

sin(x)

cos(x)

sin(x) + cos(x)

*sin(x) * cos(x)*

$$x^2 \cos(x) + \frac{\sin(x)}{3x+1}$$

Version 3

```
PROCEDURE DIFF (TREE)
   CASE TREE [SYMBOL] OF
   BEGIN
   [“^”] ANS = DIFF-EXPONL (TREE)
   [“+”] ANS = DIFF-SUM (TREE)
   [“*”] ANS = DIFF-PROD (TREE)
   [“SIN” “COS” “TAN”] = ANS = DIFF-TRIG (TREE)
   END

PROCEDURE DIFF-SUM (TREE)
MAKE-TREE (“+”,DIFF(TREE[LEFTB]),DIFF(TREE[RIGHTB]))
```

Version 3

```
PROCEDURE DIFF-PROD(TREE)
  MAKE-TREE("+",
      MAKE-TREE("*", DIFF(TREE[LEFTB]),
                     TREE[RIGHTB])
      MAKE-TREE("*", DIFF(TREE[RIGHTB]),
                     TREE[LEFTB]))
```

The New Approach …

$3x^3 + 5x + 7$

The New Approach …

$$3x^3 + 5x + 7$$

Multiply coefficient times exponent and subtract one from exponent …?

Observations about the knowledge

- It's organized around the operators.
- It's organized around nested sub-expressions
- Top-down tree descent is the natural approach
- The representation should reflect that.
- The representation should facilitate that.

Use a Natural Representation

- Conventional mathematical notation?

$$2y\sqrt{x^3 + xy(z + a)}$$

(* (* 2 y) sqrt(+ (^ x 3) (* x y (+ z a))))

- Use the pattern appropriate for the leading operator

An Implementation Approach: OOP

- Diff is a “Generic Function”
- Methods for different types of expressions
  - (defmethod diff ((n number)) 0)
  - (demethod diff ((x (eql ‘x))) 1)
  - (defmethod diff ((y symbol)) 0)
- Method for expressions does a subdispatch

```
(defmethod diff ((exp list))
    (diff-op (first exp) (rest exp)))
```

- Methods for specific operators recursively call Diff

A Small Language

- In effect we've built a *language* with the right *abstractions*:
  - Expression tree
  - Dispatching on leading operator
  - Recursive descent through the expression tree
- Operators are *independent, modular chunks* of "mathematical *knowledge*"
- Operators can be added *incrementally*
- There is an *indexing mechanism* for finding relevant operators given the structure of the current representational focus

No, really, tell it what to *know*

$$x^n \Leftrightarrow n * x^{n-1}$$

The mathematical knowledge is bidirectional
Could be used for integration as well
Even if we don't use it for that at the moment, perhaps we should preserve the opportunity to do so
More powerful pattern language for capturing the structure
More powerful matchers for enabling dispatches

Catchphrases and Punchlines

- The issue is style and pragmatics, not theory
- A program can be much more that just code.
  It can be a repository for knowledge, an environment for the development of knowledge
- Embody the reasoning, not (just) the calculation.
- Don’t tell it what to do, tell it what to know.
  - Task changes from writing a program to specifying the knowledge.
  - Task becomes debugging knowledge, not code.

Catchphrases and Punchlines

- One payoff: multiple uses of the same knowledge.

- Performance is only the beginning
  Solving the problem is only (a small) part of the job
  - Explanation
  - Learning
  - Tutoring

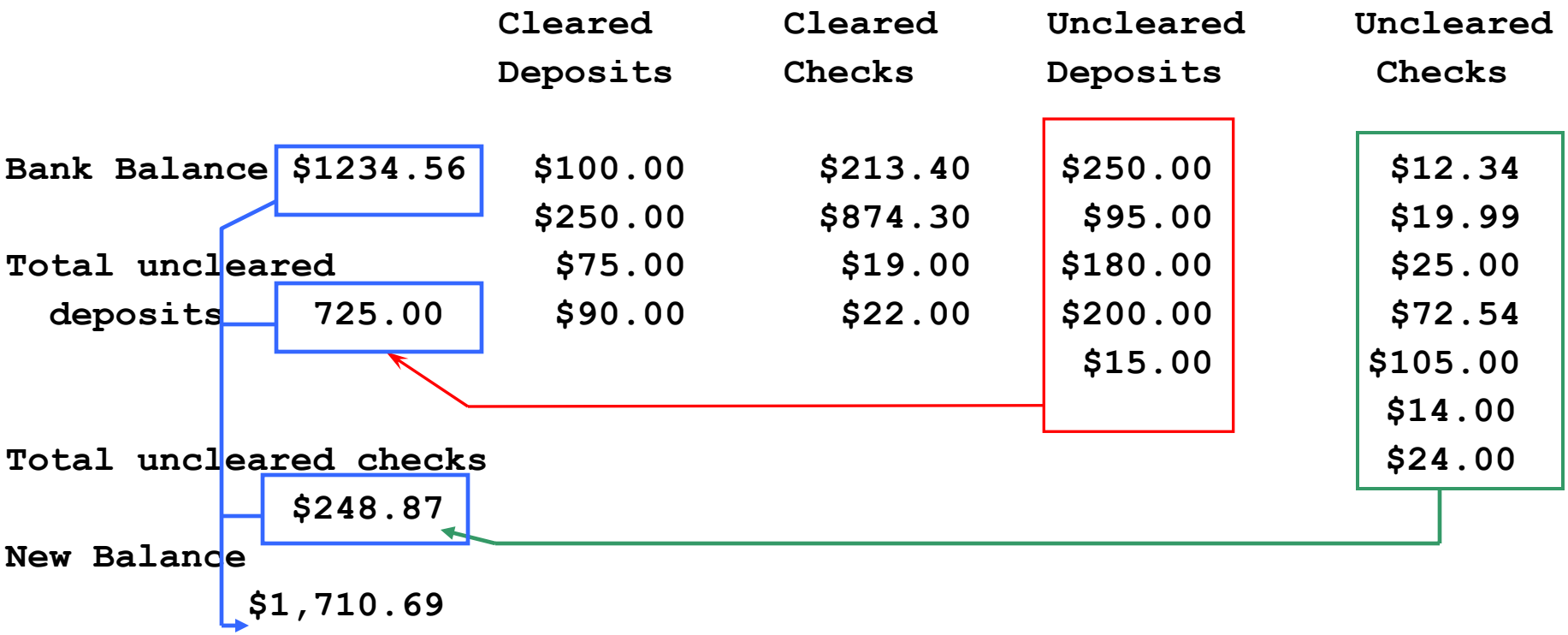
Task: Balancing Your Checkbook

```
Read StatementBalance
AdjBalance = StatementBalance
until done do {read OutstandingCheck
               AdjBalance=- OutstandingCheck}
until done do {read OutstandingDeposits
               AdjBalance=+ OutstandingDeposits}
until done do {read Fee
               AdjBalance=- Fee}
until done do {read Interest
               AdjBalance=+ Interest}
if AdjBalance = CheckBookBalance
      {print ("It balances!"); return}
else if AdjBalance > CheckbookBalance
      {print "Hey, good news."; return}
else {print "We're scrod."; return}
```

A Spreadsheet is Almost Right

- The right mindset: focus on the knowledge

The Checkbook Example



A Spreadsheet is Almost Right

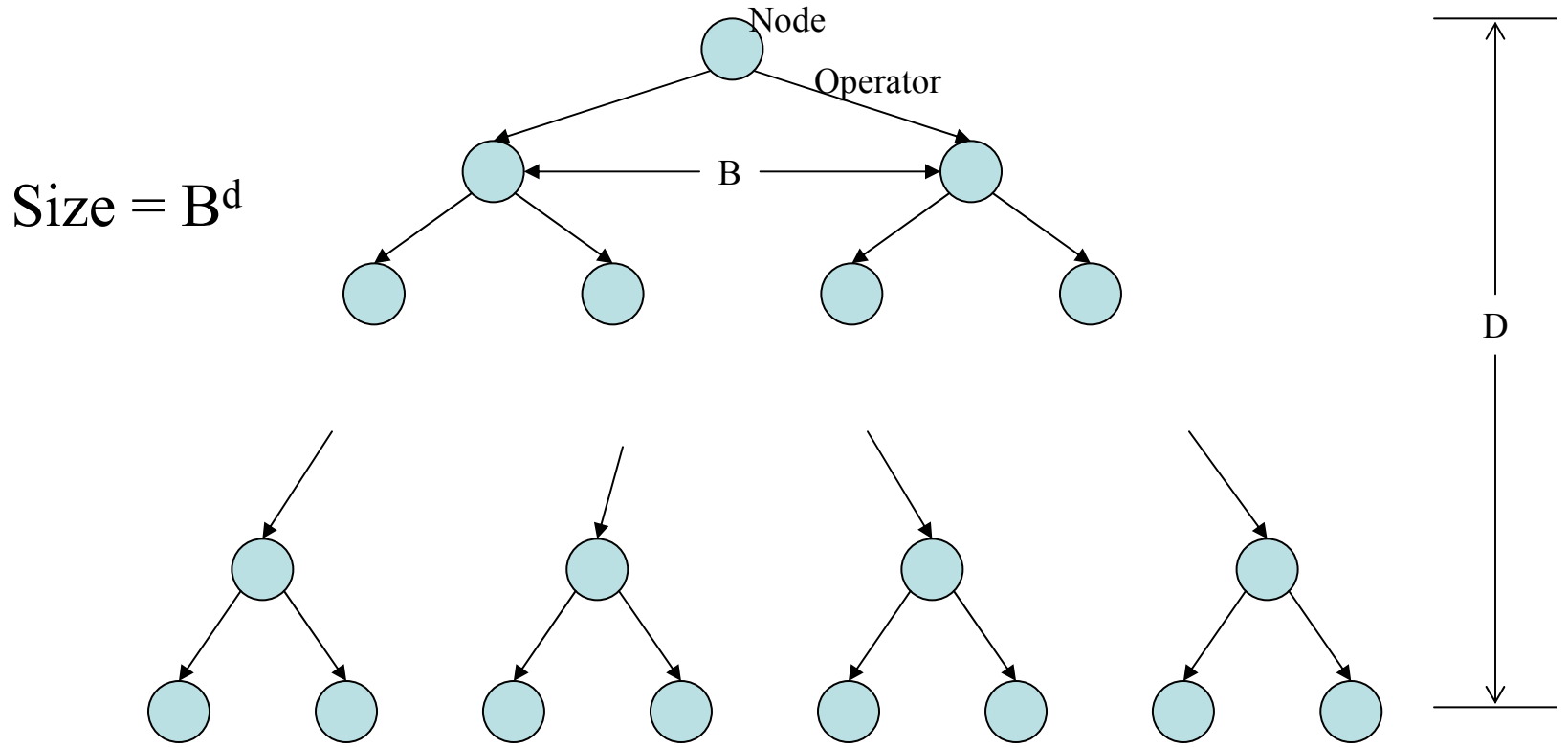
- The right mindset: focus on the knowledge
  But:
  - They are numeric and we want more
  - They have only one inference engine

- KBS as “conceptual spreadsheets”

Search Basics

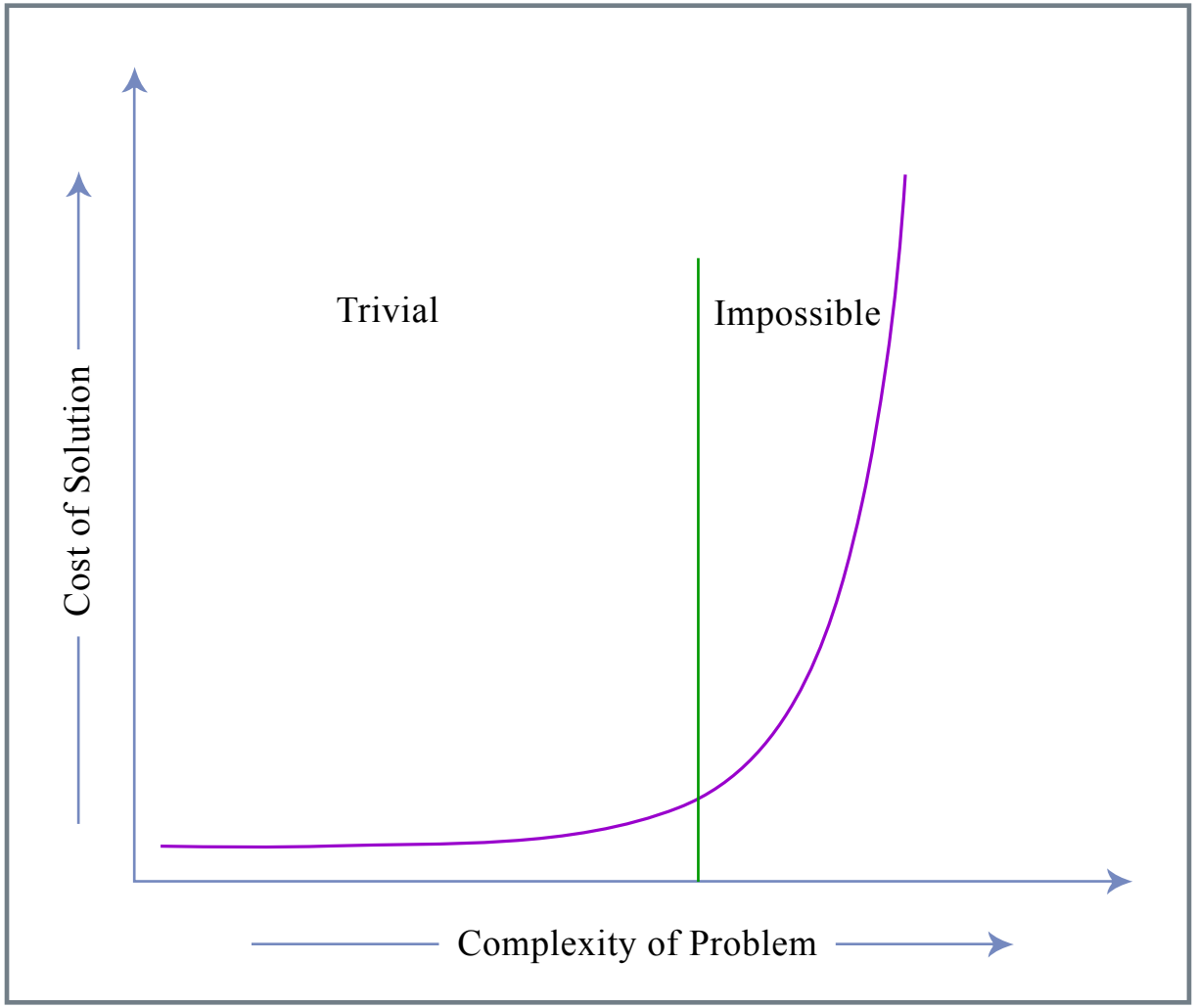
- Lecture 2, Part 2.

The Fundamental Problem: Search in a Problem Space



- B = branching factor
- D = depth

Search Spaces Grow Exponentially

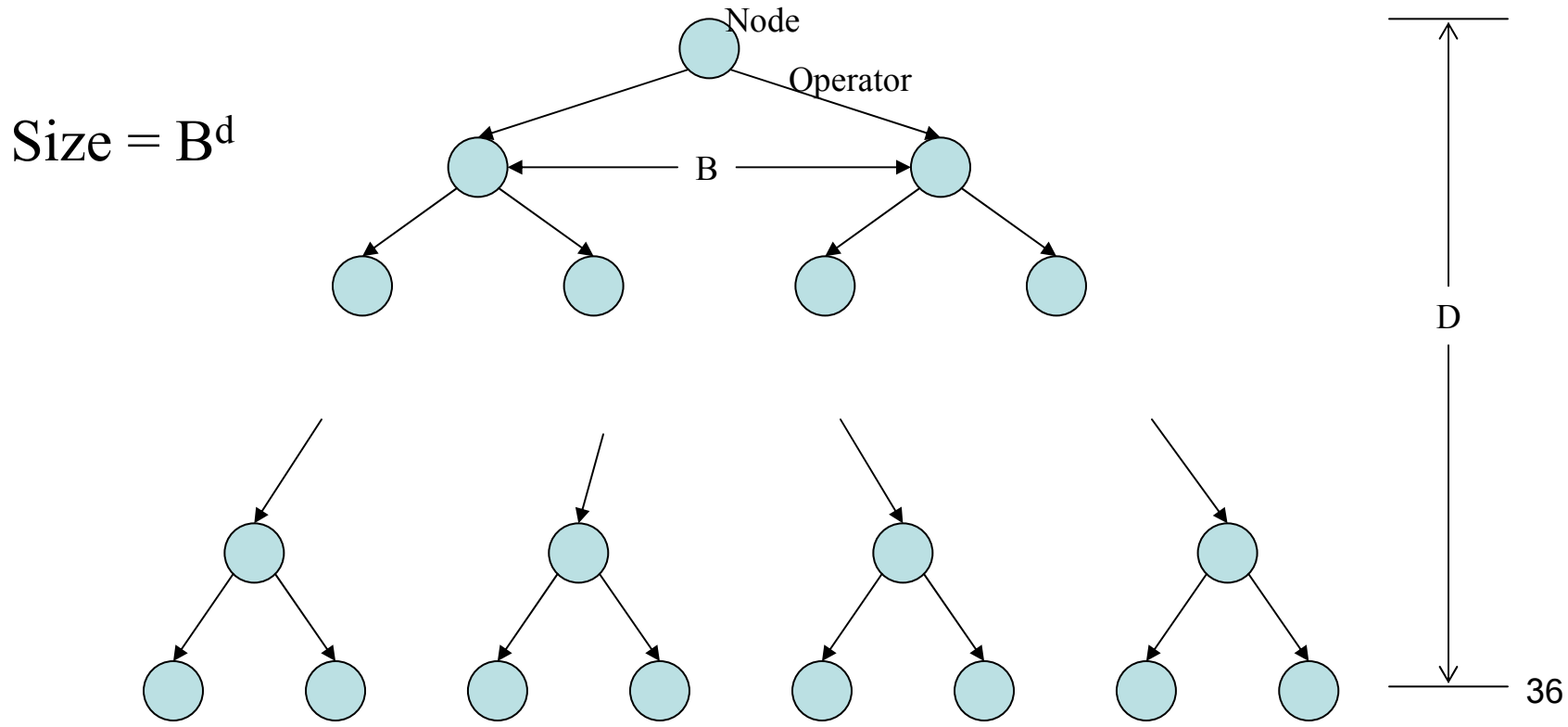


The marginal cost of slight improvement is prohibitive

Figure by MIT OCW.

The Shape of The Space

- How densely distributed are the answers?
- How uniformly distributed are the answers?
- How do answer quality and distance relate?



Depth First Search

- Go down before you go across
- Maintains focus
- Minimizes storage requirements
- Finds answer faster sometimes

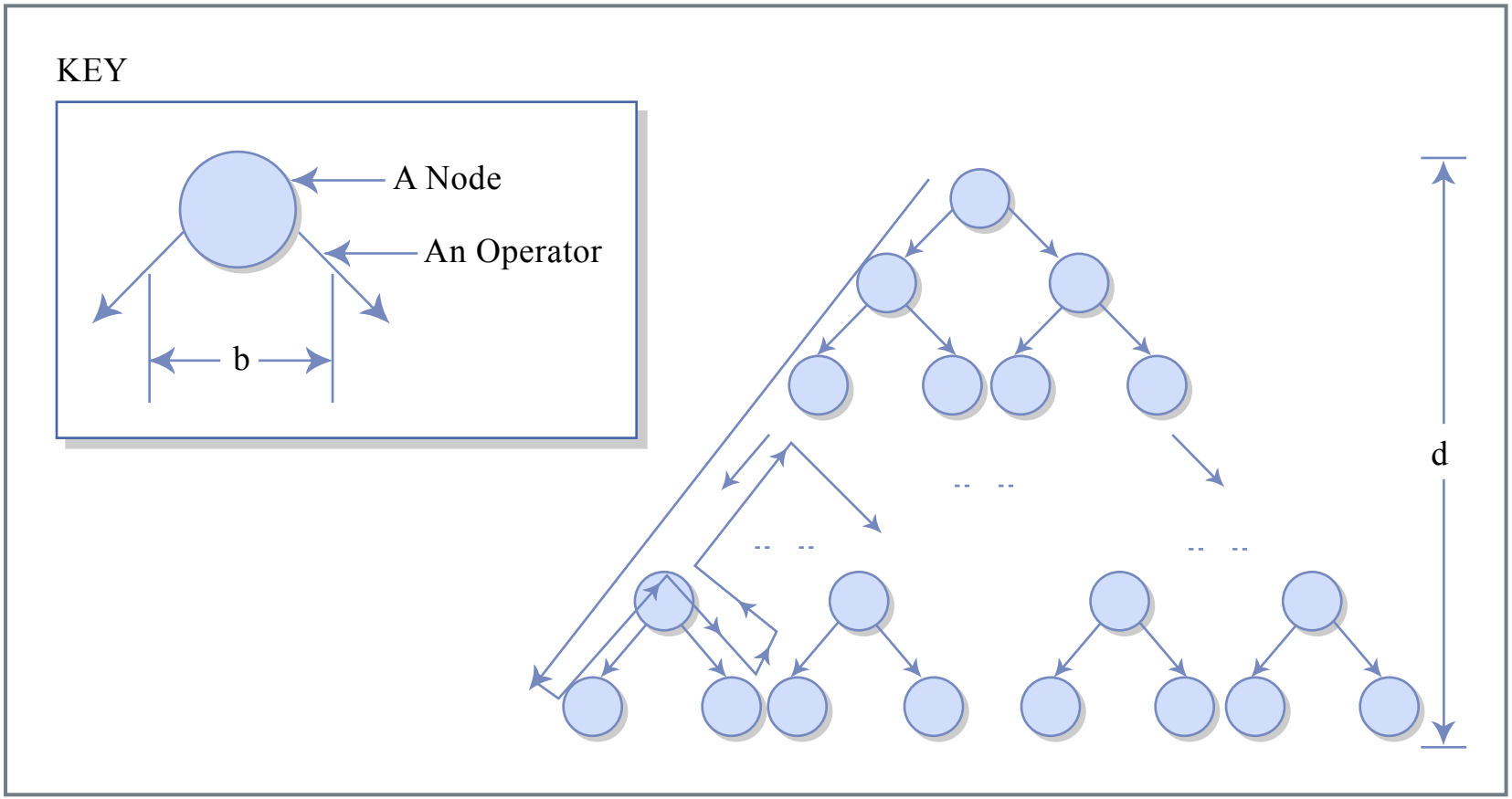
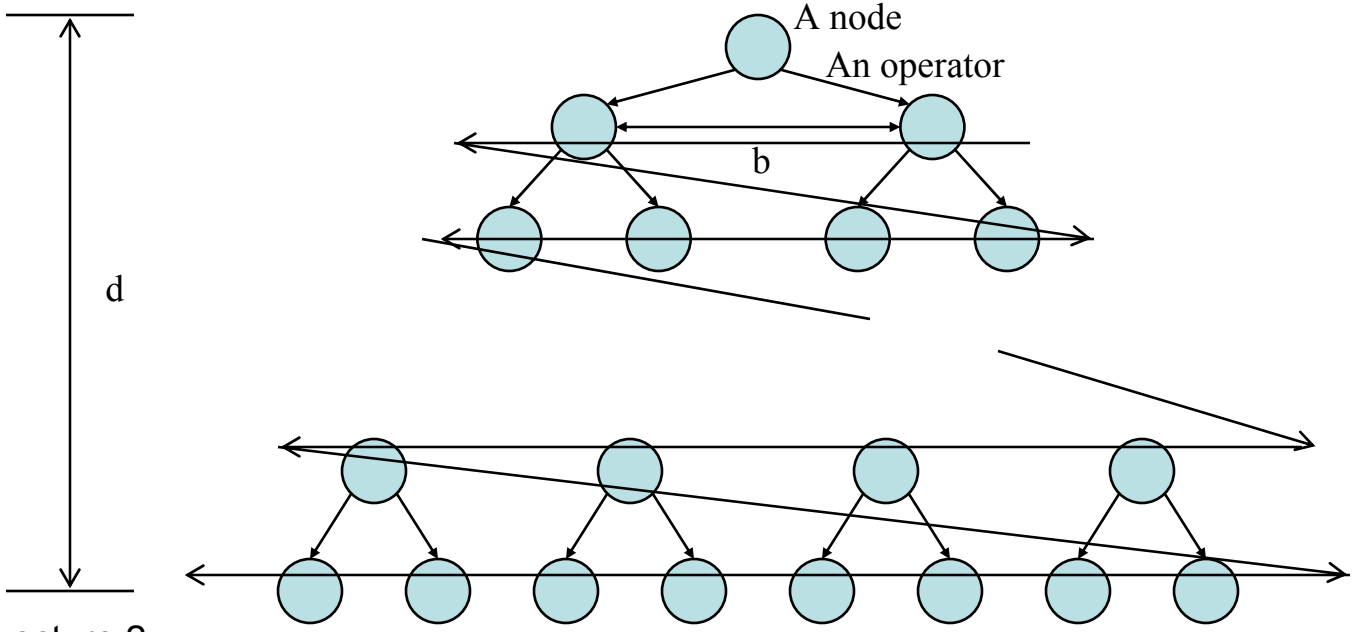


Figure by MIT OCW.

Breadth First Search

- Never gets lost on deep or infinite path
- Always finds answer if it's there
- Requires lots of storage



Best First Search

- Requires quality metric
- If metric is informed it's very quick
- Space requirements are intermediate

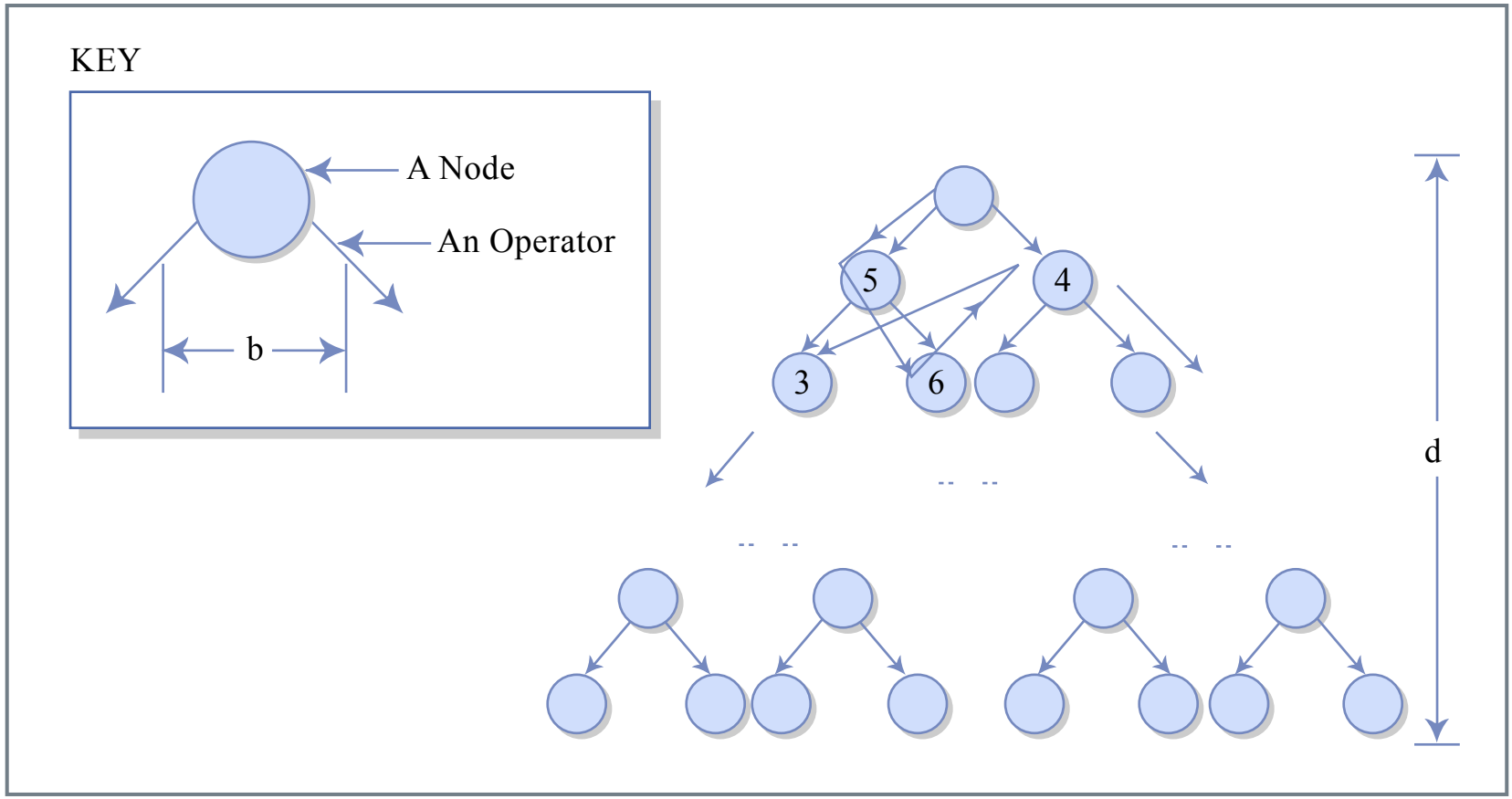


Figure by MIT OCW.

Pruning

- Throw away unpromising nodes
- Some risk that the answer is still there
- Great savings in time and space
- Breadth limited search, beam search

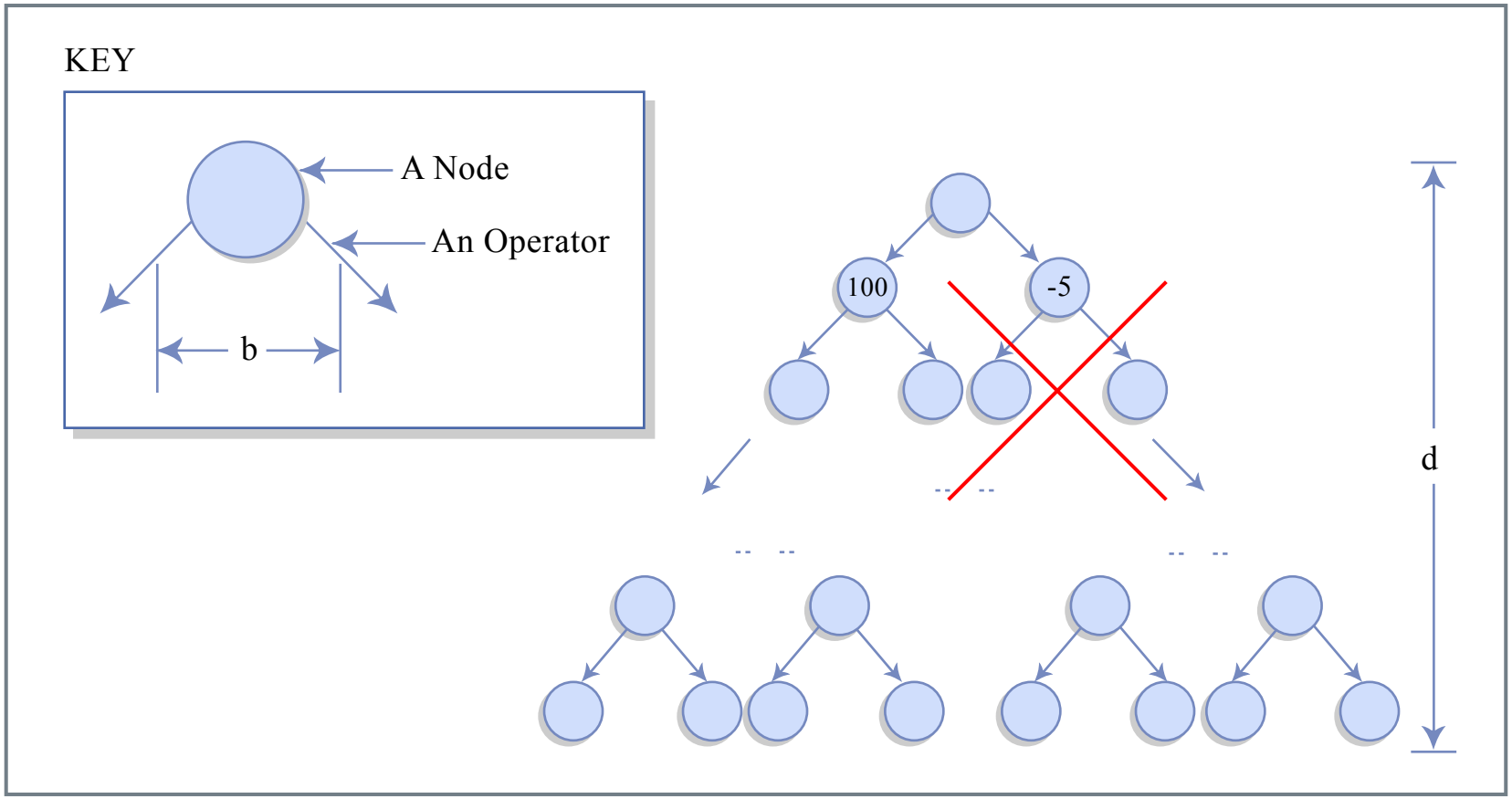


Figure by MIT OCW.

Optimum Often isn't Optimum

- In the real world things go wrong
- Robust near-optimum is usually better on average

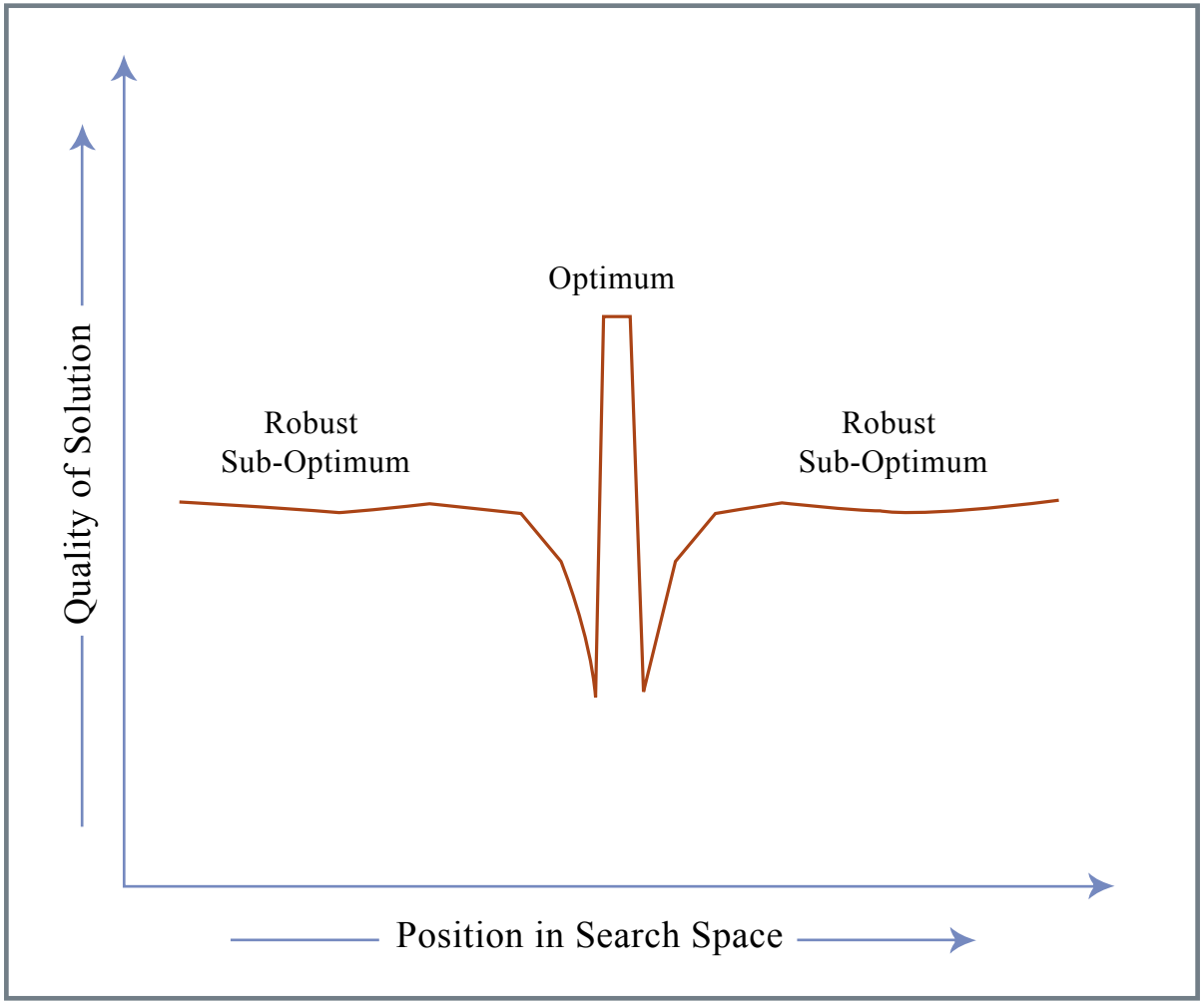
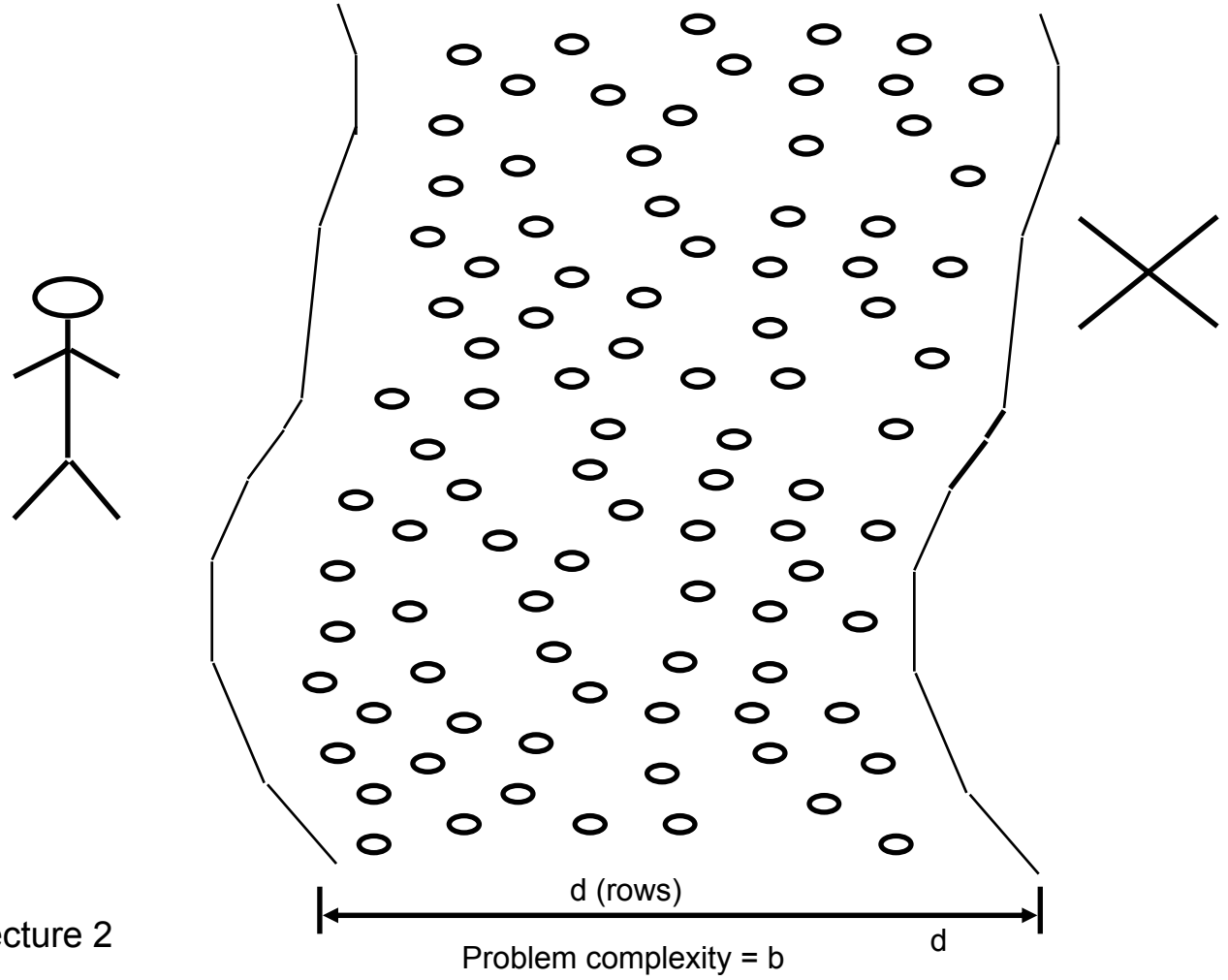


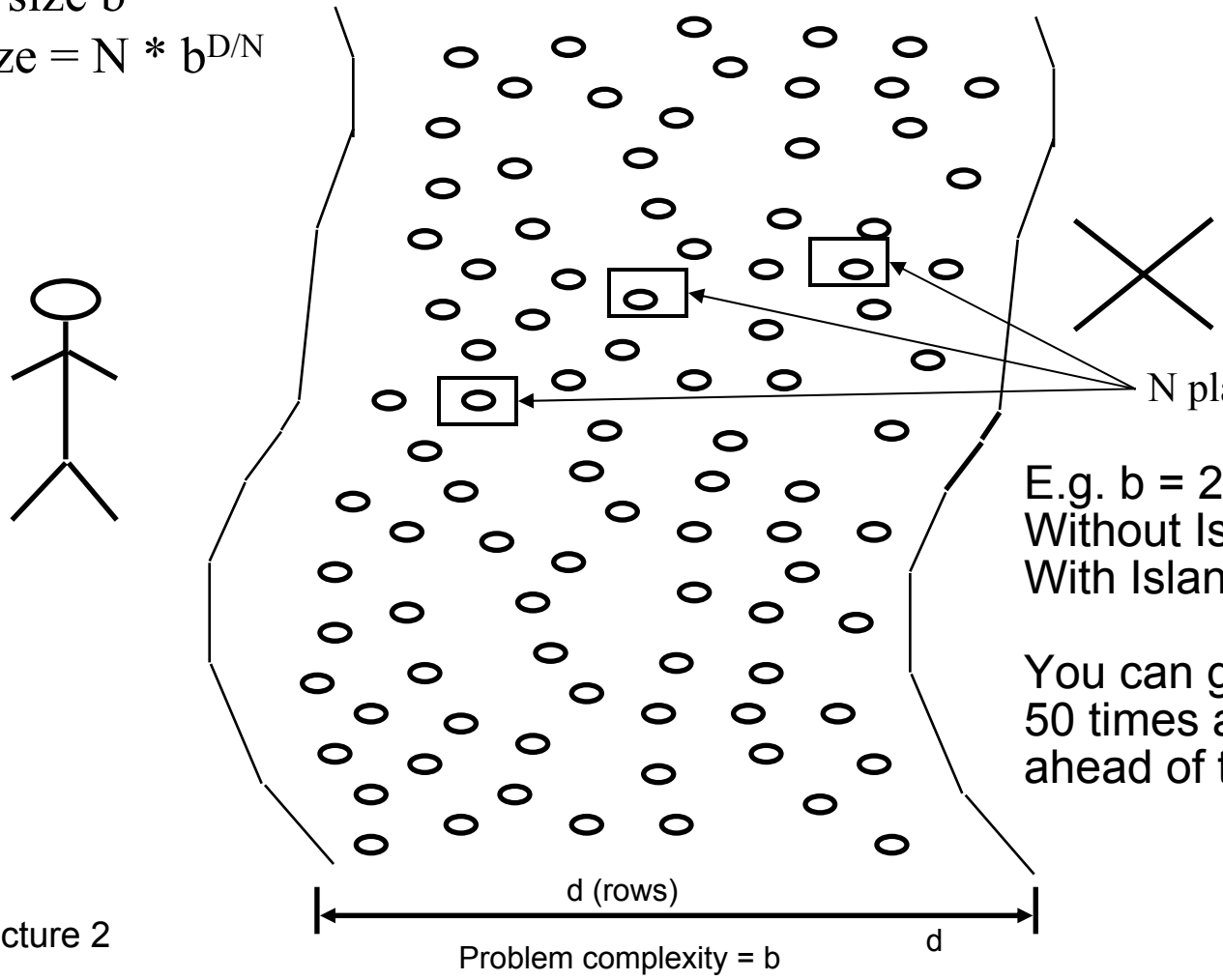
Figure by MIT OCW.

Planning Islands: The Power of Recognition



Recognizing the Form of the Problem

N subproblems
Each of depth D/N
Each of size $b^{D/N}$
Total size = $N * b^{D/N}$



E.g. b = 2, d = 10, n = 5
Without Islands: 1024
With Islands: 5 * 4 = 20

You can guess wrong 50 times and still be ahead of the game!

Summary

- All problem solving problems involve search spaces
- Search space grow intractably
- Many common algorithms for search are known

- In the Knowledge Lies the Power
  - Knowledge of a heuristic metric
  - Knowledge of planning islands
  - Knowledge of relevant abstractions
- Build representations that capture these sources of power

Version 2

```
INTEGER DEGREE, COEFF, EXPON
REAL ARRAY PROBLEM, ANSWER [1:2, 1:1000]
EXPON = 1
COEFF = 2
```

This version reads in a line of pairs of integers, coefficients and exponents, putting the coefficients in the COEFF row of P and the exponents in the EXPON row of P. Example:

$$3x^3 + 4x^2 + 5x + 7$$

results in EXPON row: 3 2 1 0
COEFF row: 3 4 5 7

Version 2

```
PROCEDURE POLY-DIFF (REAL ARRAY PROBLEM)
FOR I = DEGREE TO 1 STEP -1 DO
 BEGIN
 ANSWER [COEFF, I] = PROBLEM [EXPON, I] *
                     PROBLEM [COEFF, I]
 ANSWER [EXPON, I] = PROBLEM [EXPON, I] - 1
 END
```

But What About:

sin(x)

cos(x)

sin(x) + cos(x)

*sin(x) * cos(x)*

The Checkbook Example

		Cleared Deposits	Cleared Checks	Uncleared Deposits	Uncleared Checks
Bank Balance	$1234.56	$100.00	$213.40	$250.00	$12.34
		$250.00	$874.30	$95.00	$19.99
Total uncleared		$75.00	$19.00	$180.00	$25.00
deposits	$725.00	$90.00	$22.00	$200.00	$72.54
Total uncleared				$15.00	$105.00
checks	$248.87				$14.00
					$24.00
New Balance	**$1,710.69**				