Nondeterministic Programming in C++

Lecture 37
Sections 14.5

Robb T. Koether

Hampden-Sydney College

Wed, Nov 30, 2016

Outline

The Nondeterministic Library

- We will assume that we have available the following three functions.
  - `accept()`
  - `reject()`
  - `choose()`
- The function `accept()` outputs “accept” and exits the program.
- The function `reject()` outputs “reject” and exits the program.

The `choose()` Function

- The function `choose()` takes a set `S` as its parameter.
- `choose(S)` will return a single value from the set `S`.
- If there is a choice of values from `S` that will eventually lead the algorithm to call `accept()` and a choice that leads it to call `reject()`, then `choose()` will choose at random a value that leads to `accept()`.

The `choose()` Function

- If there is no such choice, i.e., every choice leads to `accept()` or every choice leads to `reject()`, then `choose()` will choose a random value from the set `S`.
- The execution time of `choose(S)` is O(|`S`|).

Outline

The Composites Problem

Definition (Composite Number)

A positive integer n is composite if there exist integers a and b, with $1 < a < n$ and $1 < b < n$ such that $n = ab$.

The Composites Problem

Definition (Composite Number)

A positive integer n is composite if there exist integers a and b, with $1 < a < n$ and $1 < b < n$ such that $n = ab$.

Definition (The Composites Problem)

Given a integer n, the Composites Problem (COMP) asks whether n is composite.

The Composites Problem

Definition (Composite Number)

A positive integer n is composite if there exist integers a and b, with $1 < a < n$ and $1 < b < n$ such that $n = ab$.

Definition (The Composites Problem)

Given a integer n, the Composites Problem (COMP) asks whether n is composite.

Theorem

$COMP \in \mathbf{NP}$

The Composites Problem

```
COMPOSITES(int m)
{
    int n = log2(m) + 1;
    int a = 0;
    int b = 0;
//  Generate a solution (two integers)
    for (i = 1; i <= n; i++)
    {
        a = 2*a + choose(0, 1);
        b = 2*b + choose(0, 1);
    }
//  Verify the solution
    if (m == a*b && a > 1 && b > 1)
        accept();
    else
        reject();
}
```

The Composites Problem

- Why not choose from the set of possible divisors?

```
int a = choose(2, 3, 4, ..., m - 1);
int b = choose(2, 3, 4, ..., m - 1);
```

Outline

The Subset Sum Problem

Definition (The Subset Sum Problem)

Given a set of integers $A = \{a_1, a_2, \ldots, a_n\}$ and an integer k, the Subset Sum Problem (SUBSET) asks there is a subset $B \subseteq A$ whose elements $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ add up to k.

The Subset Sum Problem

Definition (The Subset Sum Problem)

Given a set of integers $A = \{a_1, a_2, \ldots, a_n\}$ and an integer k, the Subset Sum Problem (SUBSET) asks there is a subset $B \subseteq A$ whose elements $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ add up to k.

Theorem

$SUBSET \in \mathbf{NP}$

The Subset Sum Problem

Example (The Subset Sum Problem)

- Let the set be

$$A = \{297, 820, 987, 713, 638, 545, 68, 981, 324, 559, 422, 246, 642, 485, 734, 352, 268, 431, 601, 979\}$$

and let $k = 1856$.

The Subset Sum Problem

The Subset Sum Problem

```
SUBSET(Set A, int k)
{
//  Generate a solution (a subset)
    Set B;
    for (int i = 0; i < A.size(); i++)
    {
        bool b = choose(true, false);
        if (b)
            B += i;
    }
//  Verify the solution
    int sum = 0;
    for (int i = 0; i < B.size(); i++)
        sum += A.element(i);
    if (sum == k)
        accept();
    else
        reject();
}
```

Outline

The Traveling Salesman Problem

Definition (The Traveling Salesman Problem)

Given a complete weighted graph G and an integer k, the Traveling Salesman Problem (TS) asks whether G contain a circuit that passes through every vertex and has total weight less than or equal to k?

The Traveling Salesman Problem

Definition (The Traveling Salesman Problem)

Given a complete weighted graph G and an integer k, the Traveling Salesman Problem (TS) asks whether G contain a circuit that passes through every vertex and has total weight less than or equal to k?

Theorem

$TS \in \mathbf{NP}$

The Traveling Salesman Problem

Example (The Traveling Salesman Problem)

- Let the distances between cities be

	A	B	C	D	E	F	G	H	I	J
A	0	12	33	29	53	36	58	48	73	57
B	12	0	26	41	54	41	58	53	84	69
C	33	26	0	59	33	31	36	42	88	84
D	29	41	59	0	68	47	73	53	53	28
E	53	54	33	68	0	21	5	21	73	84
F	36	41	31	47	21	0	26	12	58	64
G	58	58	36	73	5	26	0	26	78	90
H	48	53	42	53	21	12	26	0	52	65
I	73	84	88	53	73	58	78	52	0	37
J	57	69	84	28	84	64	90	65	37	0

and let $k = 487$.

The Traveling Salesman Problem

```
TS(Graph G, int k)
{
//  Generate a solution (a permutation)
    Set V = G.vertices();
    int path[G.size()];
    for (int i = 0; i < G.size(); i++)
    {
        path[i] = choose(V);
        V -= path[i];
    }
//  Verify the solution
    int dist = 0;
    for (int i = 0; i < G.size() - 1; i++)
        dist += G.weight(path[i], path[i + 1]);
    dist += G.weight(path[G.size() - 1], path[0]);
    if (dist <= k)
        accept();
    else
        reject();
}
```

Outline

Collected

Collected

To be collect on Friday, December 2.

- Section 12.1 Exercises 3, 13.
- Section 12.2 Exercise 4.
- Section 14.1 Exercises 1a, 2.
- Section 14.2 Exercises 4, 5.

Outline

Assignment

Assignment

Write C++ programs to solve the following problems nondeterministically in polynomial time using the functions `choose()`, `accept()`, and `reject()`.

(A) Solve the Hamiltonian Path Problem HAMPATH.

(B) Solve the Clique Problem CLIQ.

(C) Solve the Vertex Cover Problem VC.