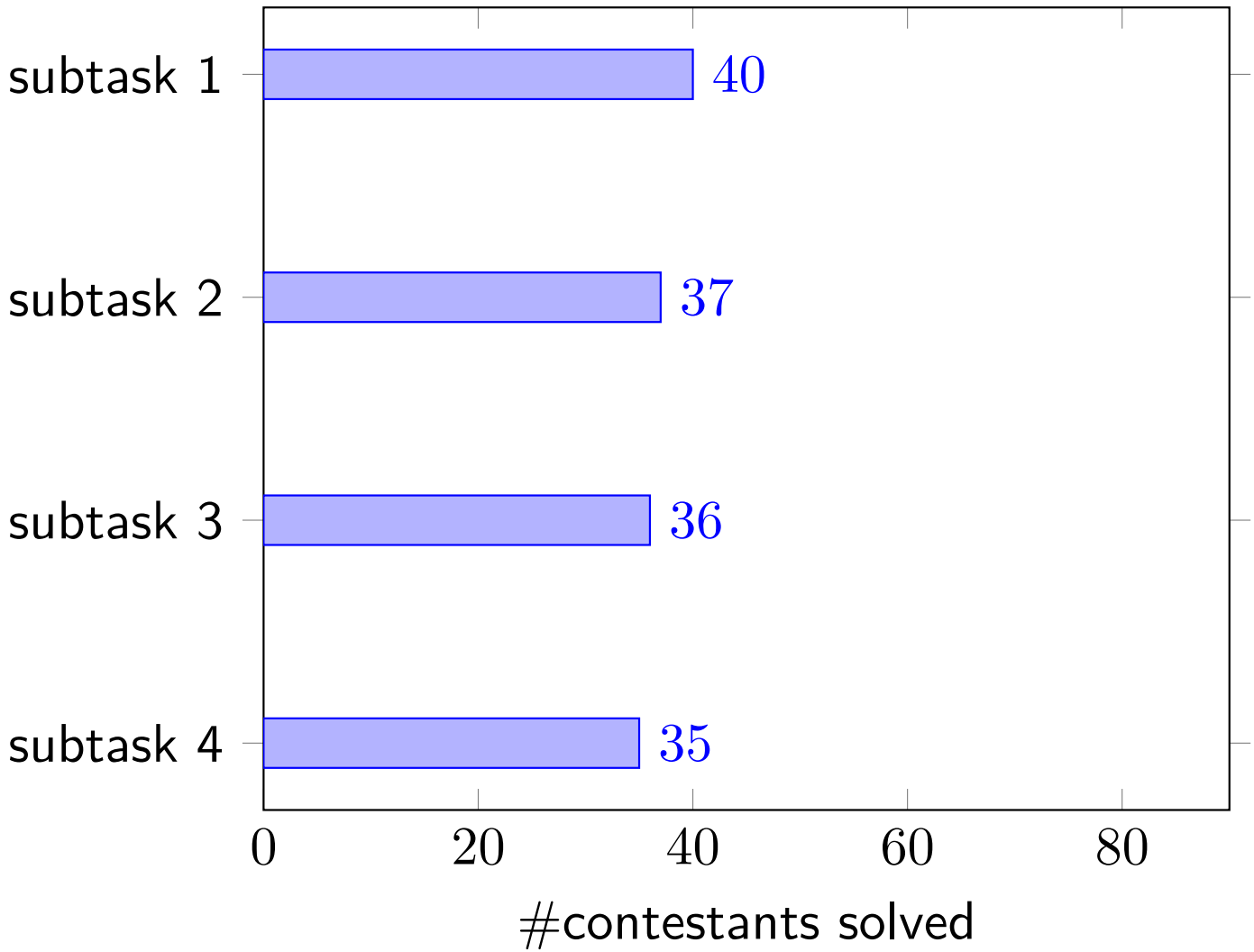
S184 Bogo Translate

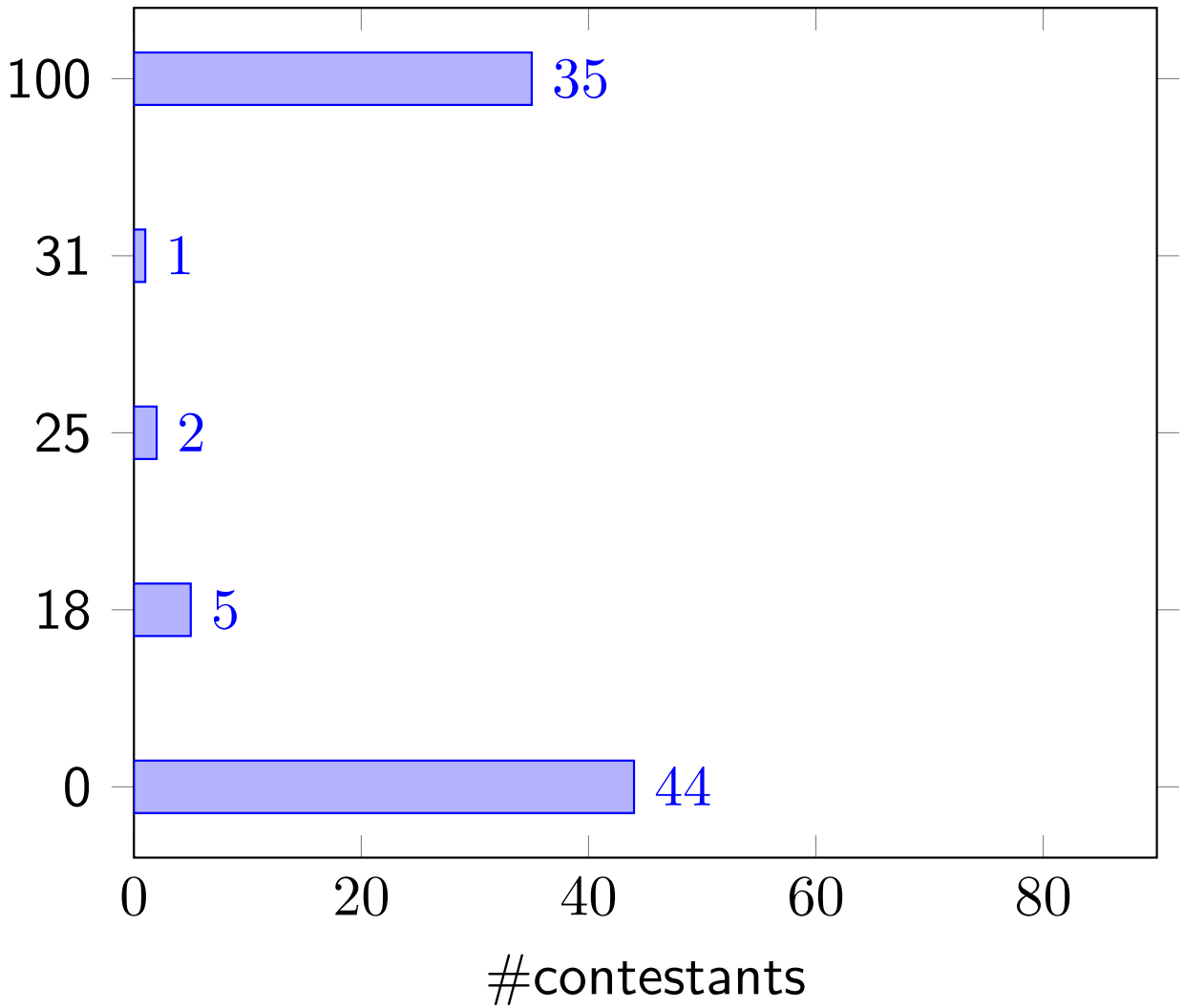
Author: Alex Tung
Speaker: Steven Lau

January 27, 2018

Statistics


subtask 1
40
subtask 2
37
subtask 3
36
subtask 4
35
0
20
40
60
80
#contestants solved

Statistics



Problem Statement

N = 3 entries in translation database

WordA	WordB
charlie	charli
i	watashiwa
am	desu

M = 1 translation task

sentence:	i am charlie
PattA:	SVO
PattB:	SOV
word-by-word:	watashiwa desu charli
final answer:	watashiwa charli desu

- $0 \leq N \leq 300$, $1 \leq M \leq 10000$
- total #words ≤ 10000
- 1 to 26 words per sentence, 1 to 15 characters per word
- Subtask 1: Every word contains only one character
- Subtask 2: PattA = PattB
- Subtask 3: Empty translation database
- Subtask 4: No additional constraints

Subtask 1: Every word contains only one character

With word being a character, we can store translation database as a character array.

WordA	WordB
d	m
r	f
m	s

```
char WordA2WordB[128];
WordA2WordB['d'] = 'm';
WordA2WordB['r'] = 'f';
WordA2WordB['m'] = 's';
```

Time complexity: $O(N)$

Subtask 1: Every word contains only one character

Word-by-word translation - with a word being a character, we don't really need to break sentence into words.

```
char sentence[1000];
gets(sentence);
int len = strlen(sentence);
for (int i = 0; i < len; i += 2)
  sentence[i] = WordA2WordB[sentence[i]];
```

Time complexity: $O(|\text{sentence}|)$

Subtask 1: Every word contains only one character

Pattern translation - build reverse lookup table

PattA	PattB
SVO	SOV

```
char PattA[16], PattB[16];
scanf("%s%s", PattA, PattB);
//strlen(PattA) == strlen(PattB) == (len + 1) / 2
int r[128];
for (int i = 0; i < len; i += 2)
  r[PattA[i / 2]] = i / 2;

//r['S'] = 0;
//r['V'] = 1;
//r['O'] = 2;
```

Subtask 1: Every word contains only one character

Pattern translation - build reverse lookup table

PattA	PattB
SVO	SOV

```
//r['S'] = 0;
//r['V'] = 1;
//r['O'] = 2;
char answer[1000];
strcpy(answer, sentence);
for (int i = 0; i < len; i += 2)
  answer[i] = sentence[r[PattB[i / 2]] * 2];
```

Time complexity: $O(|\text{sentence}|)$
Overall: $O(N + M \times |\text{sentence}|)$

Subtask 2: PattA = PattB

Cannot store translation database as character array.

WordA	WordB
charlie	charli
i	watashiwa
am	desu

```
char WordA2WordB[128];
WordA2WordB['charlie'] = 'charli';
WordA2WordB['i'] = 'watashiwa';
WordA2WordB['am'] = 'desu';
//Compilation error. Why?
```

Subtask 2: PattA = PattB

Store translation database as string arrays

WordA	WordB
charlie	charli
i	watashiwa
am	desu

```
char WordA[300][16];
char WordB[300][16];
WordA[0] = "charlie";
WordB[0] = "charli";
WordA[1] = "i";
WordB[1] = "watashiwa";
WordA[2] = "am";
WordB[2] = "desu";
```

Time complexity: $O(N \times |\text{word}|)$

Subtask 2: PattA = PattB

This time, we have to break a line into words.

```
char sentence[1000];
char word[26][16];
int words = 0;
char *p = strtok(sentence, " ");
while (p != NULL) {
  strcpy(word[words], p);
  words++;
  p = strtok(NULL, " ");
}
```

Subtask 2: PattA = PattB

Word-by-word translation:

```
for (int i = 0; i < words; i++)
  for (int j = 0; j < N; j++)
    if (strcmp(word[i], WordA[j]) == 0) {
        strcpy(word[i], WordB[j]);
        break;
    }
```

No need pattern translation

Time complexity: $O(\#\text{words} \times N \times |\text{word}|)$

Overall: $O(N \times |\text{word}| + \text{total } \#\text{words} \times N \times |\text{word}|)$

Subtask 2: PattA = PattB

What if we make use of C++ Standard Template Library?

Subtask 2: PattA = PattB (C++ STL)

Store translation database as string map.

WordA	WordB
charlie	charli
i	watashiwa
am	desu

```
map<string, string> WordA2WordB;
WordA2WordB["charlie"] = "charli";
WordA2WordB["i"] = "watashiwa";
WordA2WordB["am"] = "desu";
```

Time complexity: $O(|\text{word}| \times N \times logN)$

Where does the *logN* come from? Attend Data Structures (II).

Subtask 2: PattA = PattB (C++ STL)

Use string stream to break sentence into words.

```
string sentence;
getline(cin, sentence);
stringstream ss(sentence);
string word[26];
int words = 0;
while (ss >> word[words])
  words++;
```

Word-by-word translation:

```
for (int i = 0; i < words; i++)
  if (WordA2WordB.count(word[i]))
    word[i] = WordA2WordB[word[i]];
```

No need pattern translation

Time complexity: $O(\#\text{words} \times |\text{word}| \times logN)$

Subtask 2: PattA = PattB (C++ STL)

Lesson learnt

- C++ is very powerful
- Attend training next week: Introduction to C++

Subtask 3: Empty translation database

Pattern translation - build reverse lookup table

PattA	PattB
SVO	SOV

```
//r['S'] = 0;
//r['V'] = 1;
//r['O'] = 2;
string ordered_word[26];
for (int i = 0; i < words; i++)
  ordered_word[i] = word[r[PattB[i]]];
```

Time complexity: $O(\#\text{words} \times |\text{word}|)$

Subtask 4: No additional constraints

Just combine subtask 2 and subtask 3.

Questions?