2020 年 6 月 3 日

スーパーコン２０予選問題：文字列書き換え問題

スーパーコン２０　課題検討・審査委員会
東京工業大学・大阪大学

1.　「文字列書き換え問題」

以下では、文字列（英小文字または英大文字からなる長さ 0 以上の列）に関する問題を考える。文字列 U の長さを $|U|$ と表すことにする。

はじめに、2 つの文字列 S と T が与えられる。あなたは、以下で定義される操作（**代入**および**入れ替え**）を繰り返すことによって、文字列 S を文字列 T に変換することが可能かどうかを判定してください。また、変換可能な場合には、以下で定義される**コスト**の総和ができるだけ小さくなるような変換手順を出力してください。ただし、S および T は英大文字を含まない[1]長さ 1 以上の文字列とする。加えて、T は長さが $|S|$ 以上でかつアルファベット順に並んでいると仮定する。

（代入） 文字 x および文字列 V を選択して、「与えられた文字列 U に出現する全ての x を V に置き換える」操作を（文字 x を文字列 V に置き換える）**代入**と呼ぶ。この操作の**コスト**は $|V|+1$ とする。たとえば、文字列 $U =$ abaab に対して、文字 b を文字列 cac に置き換える代入をおこなうと文字列 acacaacac を得る。なお、この操作のコストは 4 $(= 3+1)$ である。ここで、V は「長さ 0 の文字列」でもよい。たとえば、文字列 $U =$ abaab に対して、文字 b を「長さ 0 の文字列」に置き換える代入をおこなうと文字列 aaa を得る。なお、この操作のコストは 1 である。

（入れ替え） 2 つの位置 i, j（$1 \leq i < j \leq |U|$）を選択して、「与えられた文字列 U の（左から数えて）i 番目の文字と j 番目の文字を入れ替える」操作を（i 番目の文字と j 番目の文字の）**入れ替え**と呼ぶ。この操作の**コスト**は $j - i$ とする。たとえば、文字列 $U =$ abaab に対して、3 番目の文字と 5 番目の文字の入れ替えをおこなうと文字列 abbaa を得る。なお、この操作のコストは 2 $(= 5 - 3)$ である。

入力

入力は以下の形式で与えられる：

```
S
T
```

[1]補足：（S と T を除く）変換途中の文字列においては、英大文字を含んでよい。

- 1 行目には英小文字からなる文字列 S が与えられる。
- 2 行目には英小文字からなる文字列 T が与えられる。

制約

入力データは以下の制約を満たすと仮定してよい：

- S および T の各文字は英小文字 26 種類のうちいずれかである。
- $1 \leq |S| \leq 10,000$
- $|S| \leq |T| \leq 100,000$
- T はアルファベット順に並ぶ（たとえば、`aabbb` はアルファベット順に並んだ文字列であり、`ababb` はアルファベット順に並んだ文字列ではない）。

出力

出力は、標準出力におこない、各行の末尾には改行を入れること。
S を T に変換できる場合、以下の出力形式で出力すること。

```
YES
n
P_1
⋮
P_n
```

- 1 行目には “YES” を出力する。
- 2 行目には操作の回数を表す整数 n（$0 \leq n \leq 500,000$）を出力する。
- 続く n 行は操作手順の情報を表す。n 行のうち k（$1 \leq k \leq n$）行目の各 P_k には、以下の 2 種類のうちいずれかを出力してください。ただし、各 k（$0 \leq k \leq n$）について、文字列 S からはじめて P_1 が表す操作から P_k が表す操作までを順におこなって得られる文字列を S_k と書くことにする。
  - “1 xV”：文字 x を文字列 V に置き換える代入を表す。xV は長さ $1 \leq |xV| \leq 100,001$ の文字列（V は「長さ 0 の文字列」でもよい）であり、各文字は、英大文字または英小文字 52 種類のうちいずれかである。
  - “2 i j”：i 番目の文字と j 番目の文字の入れ替えを表す。i と j は $1 \leq i < j \leq |S_k|$ を満たす整数である。

- $S_n = T$ を満たす（すなわち S からはじめて、P_1 から P_n までの操作を全ておこなった後、 T になるようにすること）。
- 各 $1 \leq k \leq n-1$ について、 $1 \leq |S_k| \leq 200,000$ を満たす。
- 上記の条件を満たさない出力については不正解として扱われる。

一方、S を T に変換できない場合には、以下の出力形式で出力すること。

```
NO
```

- 1 行目に “NO” を出力する。

予選問題での制約（次節で正確に述べる）

- 変換手順が複数存在する場合、それらのうちいずれかの変換手順を出力すれば正解とみなされる。ただし、予選ではコストの小さい解ほど有利である。
- 変換手順の解の出力は 10 回までおこなってよい。時間内に最後まで出力された解のうち最後のものが、解答として使用される。

入出力の例

例１

以下の入力を考える。

```
abaab
aaaaacccc
```

この時、たとえば以下の出力は正解として扱われる。

```
YES
3
1 bcca
2 2 6
2 3 9
```

はじめに、$S =$ abaab の文字 b を文字列 cca で置き換える代入により、文字列 accaaacca を得る。その後、2 番目と 6 番目の文字の入れ替えをおこなうことで、文字列 aacaaccca を得る。さらに、 3 番目と 9 番目の文字の入れ替えをおこな

うことで、文字列 aaaaacccc を得る。以下は上記の操作手順を表したものである（ただし各操作で変更が生じる箇所に下線が引かれている）。

$$
\begin{aligned}
S = \mathtt{a\underline{b}aa\underline{b}} &\longrightarrow \mathtt{a\underline{c}caa\underline{a}cca} && (1\ \ \mathtt{bcca})\\
&\longrightarrow \mathtt{aa\underline{c}aaccc\underline{a}} && (2\ \ 2\ \ 6)\\
&\longrightarrow \mathtt{aaaaacccc} = T && (2\ \ 3\ \ 9)
\end{aligned}
$$

なお、この時のコストの総和は $14\ (= 4 + 4 + 6)$ である。

また、以下のような出力も正解として扱われる。この時のコストの総和は $10\ (= 4 + 3 + 3)$ のため、こちらの方がより良い解である。

```
YES
3
1 bacc
2 3 6
2 4 7
```

以下は上記の出力に対応する操作手順である。

$$
\begin{aligned}
S = \mathtt{a\underline{b}aa\underline{b}} &\longrightarrow \mathtt{aa\underline{c}ca\underline{a}acc} && (1\ \ \mathtt{bacc})\\
&\longrightarrow \mathtt{aaa\underline{c}ac\underline{a}cc} && (2\ \ 3\ \ 6)\\
&\longrightarrow \mathtt{aaaaacccc} = T && (2\ \ 4\ \ 7)
\end{aligned}
$$

したがって、予選へ応募するプログラムでは、たとえば次のように出力してもよい。ここで、注意すべき点は、毎回完全な解を出力することである（たとえば、“YES” を省略しないこと）。また、解の間に空行などを入れないこと。出力の際には、参考プログラム output_sample.c の出力関数を用いるとよい。なお、解の出力途中で制限時間が過ぎた場合には、ひとつ前の解（制限時間内に完全に出力された解のうち最後のもの）が解答として使用される。

```
YES
3
1 bacc
2 2 3
2 7 8
YES
3
1 bacc
2 3 6
2 4 7
```

例２

以下の入力を考える。

```
acacaacac
aaaaaabbbb
```

この場合、S から T への変換は存在しないため “NO” を出力する。

```
NO
```

例３

以下の入力を考える。

```
bacdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

変換途中では英大文字も使用できる。たとえば以下の出力は正解として扱われる。

```
YES
3
1 aX
1 ba
1 Xb
```

また、変換途中において「長さ 0 の文字列」に置き換える代入を用いてもよい。たとえば以下の出力も正解として扱われる。

```
YES
2
1 b
1 aab
```

2. 予選問題

前節で述べた「文字列書き換え問題」を解くプログラムを作成する。

出力についての制約

- 合計 10 回まで解を出力してよい。それらのうち最後に出力されたものが解答として使用される。解を 11 回以上出力した場合には不正解と見なす。

審査方法

- 審査は、各チームの応募プログラムを、10 個の問題例に対し、各 10 秒間実行し、(i) 正解数が多い順、もし同数の場合には、(ii) コストの総和が少ない順、もし同数の場合には、(iii) 総実行時間が短い順で順位を決める。
- 10 個の問題例は、入力生成器プログラム `generator.c` に従って生成されたものを使用する（参照：予選問題補足１）。
- 最終的な予選通過者の選抜には、上記の順位の他、応募時に提出するプログラムの解説（*1）も参考にする。(*1) 解答プログラムの基本方針やアルゴリズムをＡ４で 2 ページ程度で説明した文書。チーム名`-algo.txt` もしくは、チーム名`-algo.docx` というファイルで提出すること。

実行環境

- 審査は、macOS 10.14 Mojave 搭載のコンピュータにより行い、そのコンピュータの CPU は 2.8GHz Intel Core i5 である。
- コンパイラは Homebrew GCC 9.2.0_3 を使用する。具体的には、C 言語の場合、“`gcc-9 -O2 -std=gnu11 プログラム名.c -lm`” とコンパイルし、C++ では、“`g++-9 -O2 -std=gnu++17 プログラム名.cpp`” とコンパイルする。[2]
- メモリ使用量上限を 1 ギガバイトとする。具体的には、コマンド “`ulimit -d 1000000 -m 1000000 -v 1000000`” を実行した上で審査をおこなう。
- スタック使用量上限を 8192 キロバイトとする。具体的には、コマンド “`ulimit -s 8192`” を実行した上で審査をおこなう。
- 各問題例に対して、コンパイルした実行可能プログラムを 10 秒間実行し、10 秒間のうち、最後に出力された解を使用する。プログラムは 10 秒以内に停止しなくてもよい。つまり、時間切れ強制終了となってもよい。
- 出力には、プログラム `output_sample.c` の出力関数を用いてもよい（参照：予選問題補足２）。出力部分を独自に作成してよいが、前節の例で示したような形式で出力すること。また、時間切れで強制終了した場合でも、時間内に出力したものが、確実に出力されるような注意が必要である（出力の flush を忘れずにおこなうこと）。

[2] コンパイル時に Warning などが出た場合でも、実行ファイルが作られ、実行可能な場合には（特段ペナルティを課することなく）審査を行う。

3. 級認定問題について

下記の各問題を解くプログラムを作成してください。各級ごとに 10 題の問題例で審査し、すべてで正解を出せば合格です。(なお各問について変換手順は出力せず、変換可能（YES）か不可能（NO）かのみを一度だけ出力してください。)

スーパーコン３級問題

「文字列書き換え問題」を解くプログラムを作成してください。ただし、変換手順は出力せず、変換可能（YES）か不可能（NO）か、のみを出力してください。また、入力は先述の制約に加えて以下の制約を満たす：

- S に出現する文字は全て a である。つまり S は aa...a のように、文字 a をいくつか並べた文字列である。

スーパーコン２級問題

「文字列書き換え問題」を解くプログラムを作成してください。ただし、変換手順は出力せず、変換可能（YES）か不可能（NO）か、のみを出力してください。また、入力は先述の制約に加えて以下の制約を満たす：

- $|T| \leq 30$

スーパーコン１級問題

「文字列書き換え問題」を解くプログラムを作成してください。ただし、変換手順は出力せず、変換可能（YES）か不可能（NO）か、のみを出力してください。入力に関する追加の制約はありません。

付録 A. 予選問題補足１：入力生成器

予選審査では、以下の入力生成器プログラム（generator.c）から 4 行目の疑似乱数のシード値（SEED）を「ある値」に変更したプログラムによって生成された 10 個の問題例（02_random_00.in, ..., 02_random_09.in）を用いて審査する。

```
#include <stdlib.h>
#include <stdio.h>

#define SEED 0x0000000000000001
#define MIN_S_LENG 1
#define MAX_S_LENG 10000
#define MAX_T_LENG 100000

unsigned long long xor_shift() {
```

```
  static unsigned long long x = SEED;
  x = x ^ (x << 13);
  x = x ^ (x >> 7);
  x = x ^ (x << 17);
  return x;
}

// [0, x)
unsigned long long rnd(unsigned long long x) {
  return xor_shift() % x;
}

// [l, r]
unsigned long long range_rnd(unsigned long long l, unsigned long long r)
    {
  return l + rnd(r - l + 1);
}

int comp(const void *a, const void *b) {
  return *(char *)a - *(char *)b;
}

int main () {
  for (int t = 0; t < 10; t++) {
    char file_name[64];
    sprintf(file_name, "02_random_%02d.in", t + 1);
    FILE *fp = fopen(file_name, "w");
    if (fp == NULL) {
      return 1;
    }

    int s_leng = range_rnd(MIN_S_LENG, MAX_S_LENG);
    for (int i = 0; i < s_leng; i++) {
      fprintf(fp, "%c", (char)range_rnd('a', 'z'));
    }
    fprintf(fp, "\n");

    int t_leng = range_rnd(s_leng, MAX_T_LENG);
    char *t = malloc(sizeof(char) * (t_leng + 1));
    if (t == NULL) {
      return 1;
    }
    t[t_leng] = '\0';

    for (int i = 0; i < t_leng; i++) {
      t[i] = range_rnd('a', 'z');
    }
    qsort(t, t_leng, sizeof(char), comp);
    fprintf(fp, "%s", t);

    free(t);
    fprintf(fp, "\n");

    fclose(fp);
  }

  return 0;
}
```

付録 B.　予選問題補足２：出力の方法、時間計測の方法

予選問題を解くプログラムの中で解（の候補）を出力する際には、たとえば次のような出力関数を用いるとよい（output_sample.c）。

```
/**
 * 出力の各行に対応する操作の情報を表す。
 * type = 1 のとき、代入を表す。
 * この時、S は代入の情報を表す。
 * type = 2 のとき、入れ替えを表す。
 * この時、i 番目の文字と j 番目の文字を入れ替えることを表す。
 */
struct procedure {
  int type;
  char *S;
  int i, j;
};

/**
 * 答えが YES のとき yn = 1、NO のとき yn = 0 で呼び出す。
 * n は操作の回数を表す。
 * c は操作の情報を持った procedure 型の配列。
 */
void output(int yn, int n, struct procedure *c) {
  if (!yn) {
    printf("NO\n");
  } else {
    printf("YES\n");
    printf("%d\n", n);
    int k;
    for (k = 0; k < n; k++) {
      if (c[k].type == 1) {
        printf("%d %s\n", c[k].type, c[k].S);
      } else {
        printf("%d %d %d\n", c[k].type, c[k].i, c[k].j);
      }
    }
  }
  fflush(stdout);
}
```

プログラムの中で実行経過時間を計測したい場合には、関数 gettimeofday を以下のように使うとよい（gettimeofday.c）。

```
#include <stdio.h>
#include <sys/time.h>

double get_elapsed_time(struct timeval *begin, struct timeval *end) {
  return (end->tv_sec - begin->tv_sec) * 1000
  + (end->tv_usec - begin->tv_usec) / 1000.0;
}

int main(void) {
  struct timeval t1, t2;
  gettimeofday(&t1, NULL);
  計算 (*)
```

```
  gettimeofday(&t2, NULL);
  プログラム (*) の部分の実行時間 = get_elapsed_time(&t1, &t2); /* ミリ秒 */
  return 0;
}
```