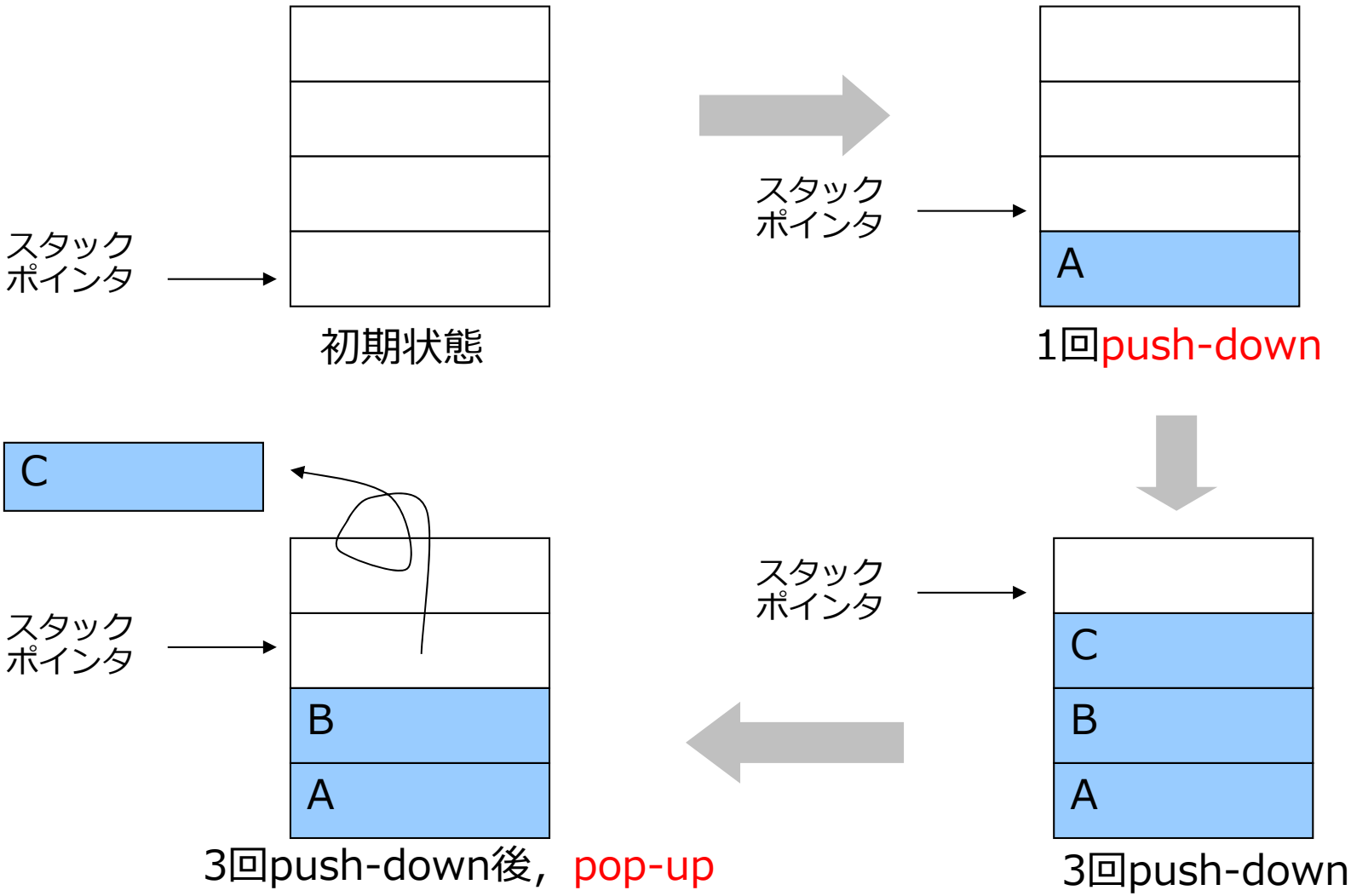
第6週
配列を使ったスタック

スタックとは

- データを保持するデータ構造
  - 後入れ先だし (Last In First Out: LIFO)
  - 箱に荷物を上に積みながら入れていき(push-down), 上から取り出す(pop-up) イメージ
- スタックポインタ
  - データの出し入れ口を指す

スタックの操作

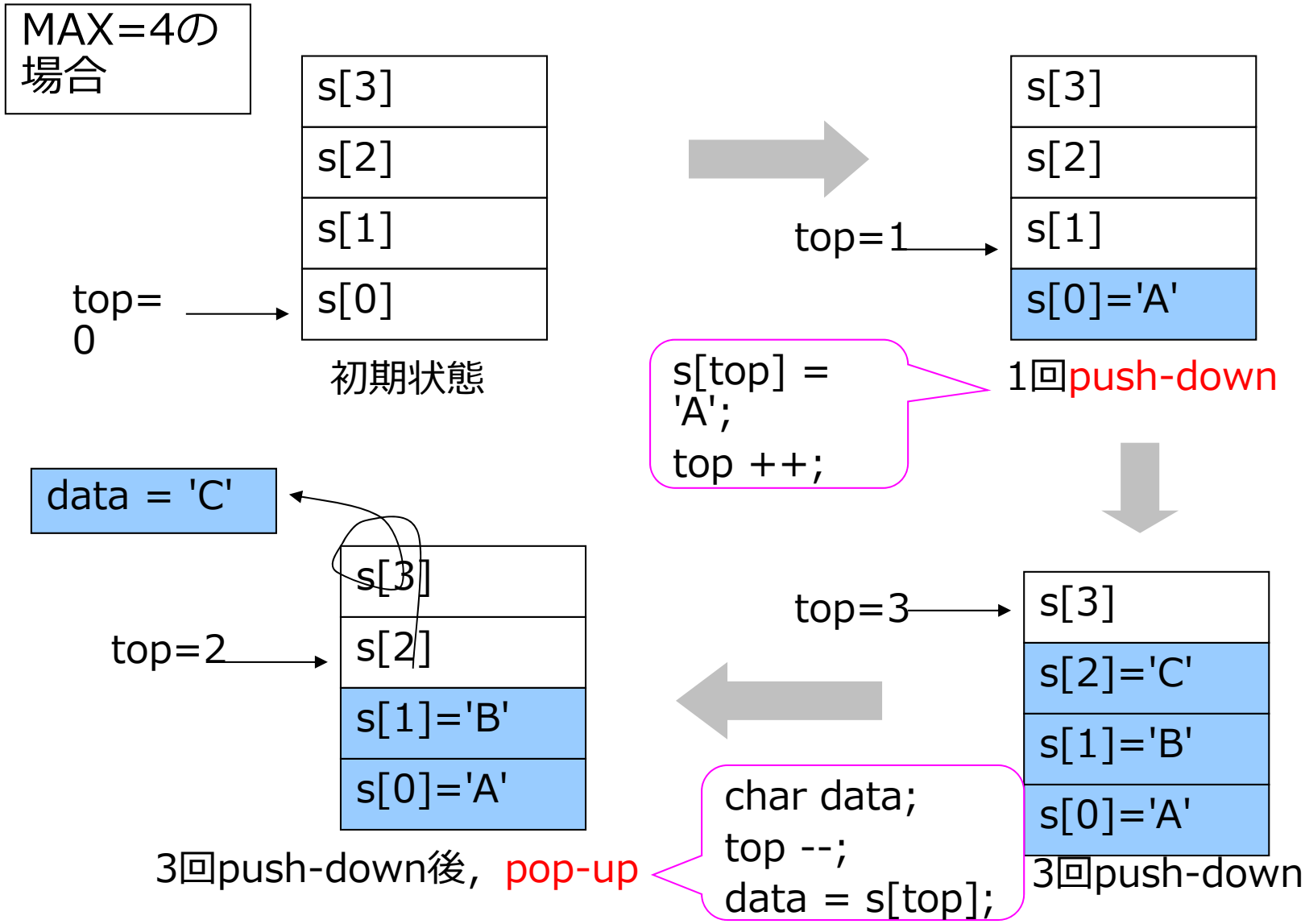

スタック
ポインタ
初期状態
スタック
ポインタ
A
1回push-down
C
スタック
ポインタ
B
A
3回push-down後，pop-up
スタック
ポインタ
C
B
A
3回push-down

配列によるスタックの表現

- 文字型データ (char型) を積んでいくスタックを考える
- char s[MAX];　　スタック用配列
- int top;　　スタックポインタ

- 初期化： top=0
- push操作: s[top] = data, top++;
- pop操作:　top--, data = s[top-1];

配列を使ったスタック操作: 必須課題6-2,3



#define文によるマクロ置換

- コンパイル時に，プログラム中の特定の文字列を，別の文字列で置換する
- 定数を定義できる
  - 同一の定数が頻出するとき，プログラムの記述や修正を簡単化できる
  - 定数の意味づけができる

```
#define MAX 30 /* 配列の大きさ */

int main(void){
    char s[MAX];     /* スタック配列 */
    int top;         /* スタックポインタ */
    char data;       /* スタックから取り出した値 */
```

ヒント集(1/3)

- データ構造とアルゴリズム教科書P.90-
- データ構造とアルゴリズムレジュメ第5回
- 課題6-1
  - 配列のサイズは「#define」で十分な大きさ確保
    - オーバフローは扱わなくてよい
  - スタックの要素は初期値として代入しておく

```
#define MAX 100

int main(void){
   char  s[MAX] ;
   int  top = 4;
}
```

  - 出力はスタックのtop-1から0まで順に出力

ヒント集（2/3）

- 課題6-2, 6-3
  - 課題6-1のプログラムを拡張
  - 関数の宣言はレジュメのものを使うこと
    - スタックの配列やtopの値は参照渡し
  - スタックの基本操作の仕方を思い出すこと
    - PUSH時（課題6-2）：
      - topに値を代入
      - topを+1
    - POP時（課題6-3）：
      - topを-1
      - topの値を出力

```
int main(void){
   char  s[MAX];
   int  top = 4;

   push('x', s, &top);
   pop(s, &top);
}
```

  - PUSH，POPの呼び出しはmain中に直接書いてよい

ヒント集(3/3)

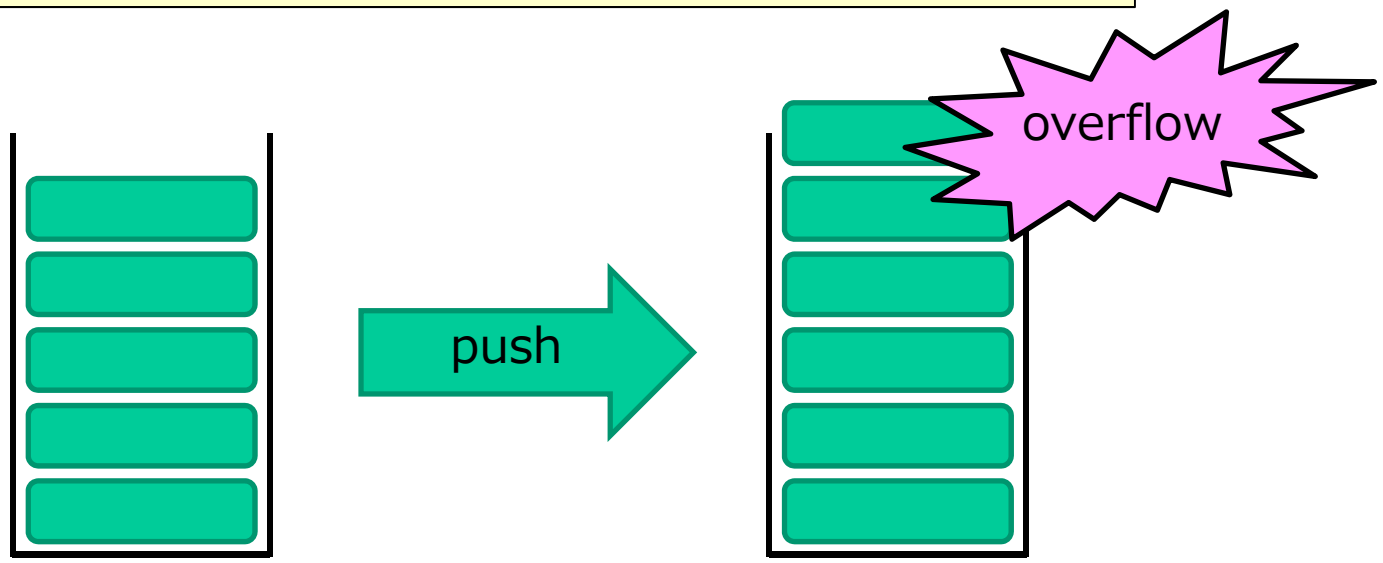
- オプション課題6-4
  - オプション課題7-4, 8-5, 8-6と近い課題
- オプション課題6-5
  - オプション課題7-5と近い課題

以降，余力のある場合

スタックのオーバーフロー

●確保したメモリ領域を超えて
pushしようとすること

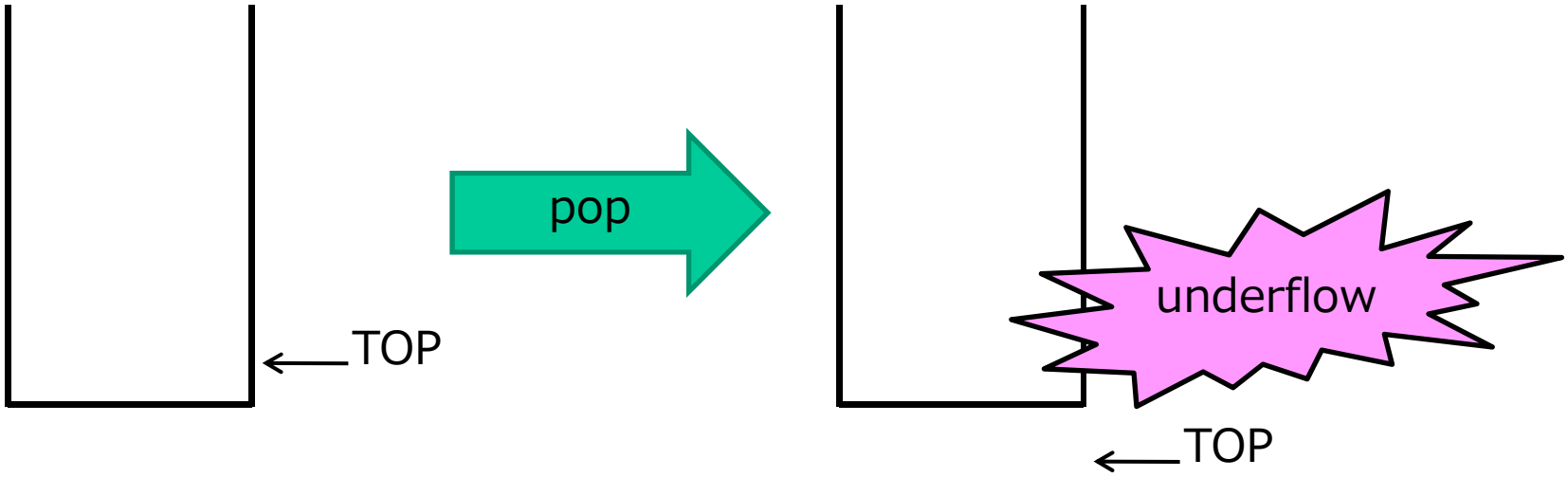
```
int stack[5];  //5つまで格納可能
```



push時には、オーバーフローが起きないように注意

スタックのアンダーフロー

●データが格納されていないスタックに対して
popしようとすること



pop時には、アンダーフローが起きないように注意

著者リスト

1. 泉　朋子 （情報コミュニケーション学科）
2. 大森 隆行（情報システム学科）
3. 原田 史子（情報システム学科）