平成２９年度後期　データ構造とアルゴリズム　期末テスト

※　各問題中のアルゴリズムを表すプログラムは，変数の宣言が省略されているなど，完全なものではありませんが，適宜，常識的な解釈をしてください．疑問があれば，挙手をして質問してください．
※　時間計算量をオーダ記法で表せという問題では，入力サイズ n を無限大に近づけた場合の漸近的な時間計算量を表せということだと考えてください．

問題１　入力サイズが n の問題を解くアルゴリズムの正確な時間計算量が次のような場合，それぞれの漸近的な時間計算量をオーダ記法で表せ．

（１）3　　（２）$2n$　　（３）$4n^2+3n$

（４）$5n^3-n-2$　　（５）$2\sqrt{n}+3n-1$　　（６）$3\log_{10}n+2n+1$

（７）$3\sqrt{n}+4\log_{10}n-2$　　（８）$2n\log_{10}n+5n+1$　　（９）$3n\log_{10}n+2n^2-4$

(10) $n^{100}+2^n$　　(11) $2^n+n!$　　(12) $n!+n^{100}$

問題２　次のプログラムで表される，入力サイズが n の問題を解くアルゴリズムの時間計算量をオーダ記法で表せ．（２）の n は 2 のべき乗の数（整数 m を用いて 2^m と表すことができる数）とする．

（１）

```
x = 0;
for( i = 1; i <= n; i++ ) {
    x = x + i;
}
```
x の値を出力する;

（２）

```
x = n;
while( x > 1 ) { x = x / 2; }
```
x の値を出力する;

（３）

```
for( i = 1; i <= n; i++ ) { x[i] = 0; }
for( i = 1; i <= n; i++ ) {
    x[i] = i;
    for( j = i+1; j <= n; j++ ) {
        x[i] = x[i] + j;
    }
}
for( i = 1; i <= n; i++ ) {
    x[i]の値を出力する;
}
```

問題３　サイズ n の配列 S[0], …, S[n-1]を用いて，次の２つの関数でスタックを実現する．下の問いに答えよ．

```
/* スタック S に対して，データ x を格納する */
push( S, x ) {
    top = (a);
    if( (b) ) {
        "オーバーフロー"と出力;
    }
    else {
        S[top] = x;
    }
}
```

```
/* スタック S からデータを取り出し，そのデータを出力する */
pop( S ) {
    if( (c) ) {
        "アンダーフロー"と出力;
    }
    else {
        S[top]の値を出力;
        top = (d);
    }
}
```

（1）関数中の空欄を埋めよ．

（2）スタックを表す配列 S を空に初期化する場合，変数 top をどのような値にすればよいか答えよ．

（3）空のスタック S に対して以下の操作を順番に実行した．

push(S, 4) → push(S, 3) → push(S, 8) → pop(S)
→ pop(S) → push(S, 7) → push(S, 1) → pop(S)

（３－１）１，２，３回目の pop で出力される値をそれぞれ答えよ．

（３－２）配列のサイズが n = 4 の場合，全ての操作の終了後に配列の各要素に入っている値を答えよ．
ただし，pop で出力した値は配列に残らないものとし，値が入っていない場合には × と答えよ．

問題４ サイズ n の配列 Q[0], …, Q[n-1]を用いて，次の２つの関数でキューを実現する．下の問いに答えよ．

```
/* キューQ に対して，データ x を格納する */
enqueue( Q, x ) {
    Q[right] = x;
    right = (a) ;
    if( right == n ) { (b) }
    if( left == right ) {
        "オーバーフロー"と出力;
    }
}
```

```
/* キューQ からデータを取り出し，そのデータを出力する */
dequeue( Q ) {
    if( left == right ) {
        "アンダーフロー"と出力;
    }
    else {
        Q[left]の値を出力;
        left = (c) ;
        if( left == n ) { (d) }
    }
}
```

（１）関数中の空欄を埋めよ．

（２）キューを表す配列 Q を空に初期化する場合，変数 left と right をどのような値にすればよいか答えよ．

（３）空のキューQ に対して以下の操作を順番に実行した．

enqueue(Q, 4) → enqueue(Q, 3) → enqueue(Q, 8) → dequeue(Q)
→ dequeue(Q) → enqueue(Q, 7) → enqueue(Q, 1) → dequeue(Q)

（３－１）１，２，３回目の dequeue で出力される値をそれぞれ答えよ．

（３－２）配列のサイズが n = 4 の場合，全ての操作の終了後に配列の各要素に入っている値を答えよ．ただし，dequeue で出力した値は配列に残らないものとし，値が入っていない場合には × と答えよ．

問題５　次のプログラムは，昇順（小さな値から大きな値の順）の n 個のデータ D[0], ..., D[n-1]から“**２分探索法**”によって値 x を探索する.

```
void binarysearch(
    double D[],        // データ D[0], ..., D[n-1]
    int n,             // データの個数
    double x           // 探索する値
)
{
    left = 0;    right = n - 1;    mid = ( left + right ) / 2;
    while( left < right ) {
        if( D[mid] == x ) { break; }
        else if( D[mid] < x ) { left = (a); }
        else { right = (b); }
        mid = (c);
    }
    if( D[mid] == x ) { D[mid]を出力; }
    else {    (d)    }
}
```

（１）空欄(a)～(c)を埋めよ.

（２）(d)の箇所が実行されるのはどのような場合か答えよ.

（３）このアルゴリズムの時間計算量をオーダ記法で表せ.

問題６　次のプログラムは**“ハッシュ法”**を実現するものであり，関数 hash_input はハッシュ関数 hash によるデータの格納，関数 hash_search はデータの探索を行う．各関数の引数については，n ≦ m とし，与えられた n 個のデータを D[0], ..., D[n-1]，ハッシュ関数によりデータを格納する大きさ m の配列を H[0] ..., H[m-1]，探索する値を x とする．また，NO_VAL は配列 H 中にデータが格納されていないことを示す値であり，データとして格納される値とは異なるものとする．下の問いに答えよ．

```
void hash_input(
    int n, int m, int D[], int H[] )
{
    // 配列 H の初期化
    for( i = 0; i < m; i++ ) {
        H[i] = NO_VAL;
    }
    // データの格納
    for( i = 0; i < n; i++ ) {
        k = hash( m, D[i] );
        while( H[k] != NO_VAL ) {
            k = ( k + 1 ) % m;    // (A)
        }
        H[k] = D[i];
    }
}
```

```
int hash_search(
    int m, int H[], int x )
{
    k = hash( m, x );
    while( H[k] != NO_VAL ) {
        if( H[k] == x ) {
            return( k );    // (B)
        }
        k = ( k + 1 ) % m;    // (A)
    }
    return( -1 );    // (C)
}

int hash( int m, int x )
{
    return( x % m );
}
```

（1）データの個数が n = 8 で，配列 D[0], …, D[7]に下記の値がその順序で入っているとする．また，配列 H の大きさを m = 12 とする．このとき，関数 hash_input の実行後に配列 H[0], …, H[11]に格納されている値を書け．値が格納されていない場所には × を書け．

(a) 1, 3, 7, 9, 15, 19, 23, 27

(b) 3, 9, 19, 27, 1, 7, 23, 15

(c) 3, 19, 1, 27, 8, 5, 15, 7

（2）プログラム中の２箇所の(A)で　% m　によってｍによる除算の余りを求める理由を書け．

（3）関数 hash_search の(B)で返り値とされる値 k は何かを書け．

（4）関数 hash_search の(C)で返り値が -1 とされるのはどのような場合かを書け．

問題７　次のプログラムは，配列 D[0], ..., D[n-1]中の n 個のデータを**“選択ソート”**によって昇順に並べ変える．関数 swap は２つの引数のデータを交換するものである．下の問いに答えよ．

```
void selectionsort( int n, double D[] )
{
    for( i = n-1; i > 0; i-- ) {
        max = D[0];
        max_index = 0;
        for( j = 1; j <= i; j++ ) {
            if( D[j] >= max ) {
                max = D[j];
                (a)
            }
        }
        swap( &(D[max_index]), &( (b) ) );
    }
}
```

（1）空欄を埋めよ．

（2）データの個数が n = 5 で，ソート前の初期状態として配列 D[0], …, D[4]に 9, 1, 5, 3, 7 がこの順序で入っていたとする．変数 i による外側の for ループの繰り返しが，変数 i が 4, 3, 2, 1 の値で順に実行されるとき，各値での実行が終了した直後の配列 D 中のデータを書け．

（3）このアルゴリズムの時間計算量をオーダ記法で表せ．

問題８　次のプログラムは，配列 D[0], ..., D[n-1]中の n 個のデータを**“挿入ソート”**によって昇順に並べ変える．下の問いに答えよ．

```
void insertionsort( int n, double D[] )
{
    for( i = 1; i < n; i++ ) {
        x = D[i];
        j = i;
        while( ( D[j-1] > x ) && ( (a) ) ) {
            (b)
            j = j - 1;
        }
        D[j] = x;
    }
}
```

（1）空欄を埋めよ．

（2）データの個数が n = 5 で，ソート前の初期状態として配列 D[0], …, D[4]に 9, 1, 5, 3, 7 がこの順序で入っていたとする．変数 i による外側の for ループの繰り返しが，変数 i が 1, 2, 3, 4 の値で順に実行されるとき，各値での実行が終了した直後の配列 D 中のデータを書け．

（3）このアルゴリズムの時間計算量をオーダ記法で表せ．

問題９　次のプログラムは，**“ヒープ”**にデータを追加する関数 push_heap とヒープから最大値を取り出す関数 delete_maximum である．ヒープの２分木を配列 T で表し，変数 size はそのサイズである．また，関数 swap は２つの引数のデータを交換する．下の問いに答えよ．

```
void push_heap( double T[], /* ヒープを表す配列 T[1], ..., T[size] */
                double x /* 追加するデータ */ )
{
    size++;
    T[size] = x; // データを最後に追加
    k = size;
    while( ( k > 1 ) && ( [ (a) ] ) ) {
        swap( &(T[k]), &(T[k/2]) );  k = [ (b) ];
    }
}

double delete_maximum( double T[] /* ヒープを表す配列 T[1], ..., T[size] */ )
{
    T[1]を出力;
    T[1] = T[size];  T[size]を空にする; // 最後のデータを根に移動
    size--;  k = 1;
    while( 2*k <= size ) { // 子を持つかどうかを判定
        if( 2*k == size ) {
            if( [ (c) ] ) {
                swap( &(T[k]), &(T[2*k]) );  k = [ (d) ];
            }
            else { アルゴリズムを終了; }
        }
        else {
            if( [ (e) ] ) { big = 2*k; }
            else { big = 2*k+1; }
            if( T[k] < T[big] ) {
                swap( &(T[k]), &(T[big]) );  k = [ (f) ];
            }
            else { アルゴリズムを終了; }
        }
    }
}
```

（1）空欄を埋めよ．

（2）ヒープのサイズが size = 10 であり，配列 T[1], …, T[10]の値が

39, 23, 24, 17, 6, 1, 2, 9, 11, 5

である場合，次のそれぞれについて答えよ．

（２－１）ヒープにデータ x = 31 を追加するために関数 push_heap を実行する．この実行後に T[1], …, T[11]に格納されている値をその順序で答えよ．

（２－２）ヒープから最大値を取り出す関数 delete_maximum を実行する．この実行後に T[1], …, T[9]に格納されている値をその順序で答えよ．

問題１０ 次のプログラムは，mergesort(D, 0, n-1)を実行することで，配列 D[0], ..., D[n-1] 中の n 個のデータを**“マージソート”**によって昇順に並べ変える．なお，プログラム中の変数の宣言は省略している．下の問いに答えよ．

```
void mergesort(
    double D[],        // データ D[left], ..., D[right]
    int left,          // ソートの対象とする配列 D の左端の位置
    int right          // ソートの対象とする配列 D の右端の位置
)
{
    mid = ( left + right ) / 2;                                    // (a)
    if( left < mid ) { mergesort( D, left, mid ); }                // (b)
    if( mid+1 < right ) { mergesort( D, mid+1, right ); }          // (c)
    merge( D, left, mid, right );
}
void merge(
    double  D[],    // データ D[left], ..., D[right]
    int     left,   // ソートの対象とする配列 D の左端の位置
    int     mid,    // ソートの対象とする配列 D の中央の位置
    int     right   // ソートの対象とする配列 D の右端の位置
)
{
    x = left;    y = mid + 1;
    for( i = 0; i <= right-left; i++ ) {
        if( x == mid+1 ) { M[i] = D[y]; y++; }              // (d)
        else if( y == right+1 ) { M[i] = D[x]; x++; }       // (e)
        else if( D[x] <= D[y] ) { M[i] = D[x]; x++; }       // (f)
        else { M[i] = D[y]; y++; }                          // (g)
    }
    for( i = left; i <= right; i++ ) { D[i] = M[i]; }
}
```

（1）データの個数が n = 10 で，配列 D[0], …, D[9]に

35, 21, 4, 49, 55, 19, 12, 32, 24, 42

がこの順序で入っている状態で，関数 mergesort が left = 0, right = 9 で呼び出されたとする．

（1－1）(a)で mid に代入される値を答えよ．

（1－2）(b)の関数 mergesort の実行後の D[left], …, D[mid]中の値をその順序で答えよ．

（1－3）(c)の関数 mergesort の実行後の D[mid+1], …, D[right]中の値をその順序で答えよ．

（2）関数 merge に渡される D[left], …, D[mid]を左データ列，D[mid+1], …, D[right]を右データ列と呼ぶこととする．(d)から(g)のそれぞれが実行されるのはどのような場合か，左データ列と右データ列という用語を用いて答えよ．

問題１１　表１に示す３種類のコーヒー豆が入った袋があり，これらは分割できない．これらを重さの限界が 3 kg のバッグに入れるとき，価格の総和を最大にする豆袋の入れ方を求めたい．この問題を動的計画法によって解くため，次の３つの部分問題に分割する．

・ 部分問題１　豆１だけがある．
・ 部分問題２　豆１と豆２がある．
・ 部分問題３　豆1，豆2，豆3の全てがある．

表１　コーヒー豆の重さと価格

番号 i	重さ w_i (kg)	価格（ドル）
1	2	30
2	1	20
3	3	40

部分問題１は，豆１を入れるか入れないかのいずれかを解とする．その後，部分問題２は部分問題１の解に豆２を追加して解くことができ，部分問題３は部分問題２の解に豆３を追加して解くことができる．この解法は表２を利用して実現できる．表２の各マスには，部分問題 i（ = 1，2，3 ）ごとに，重さの和の限界が w である場合に価格の総和が最大となる豆の組み合わせを**（ 価格の総和，{入れる豆の番号} ）**として記録する．これを次の方針に従って行う．

> 部分問題 i について，重さの和の限界が w である場合の解は，次の入れ方のうちで価格の総和が高いほうである．
> ① 豆 i は入れず，部分問題 i － 1 に対する重さの和の限界が w である場合の解となる入れ方
> ② 部分問題 i － 1 に対する重さの和の限界が w － w_i である場合の解となる入れ方に重さ w_i の豆 i を加える入れ方

表２は，部分問題１で豆１を入れない場合と入れる場合に相当する２箇所だけが埋められている．記号 ∅ は何も入れないことを表す．上記の方針に従い，表２の空白のマスを埋め，その結果から価格の総和を最大にする豆袋の入れ方を答えよ．

表２　動的計画法のための記録表

		重さの和の限界 w (kg)			
		0	1	2	3
部分問題	1	(0, ∅)		(30, {1})	
	2				
	3				

平成２９年度後期　データ構造とアルゴリズム　期末テスト　解答用紙

平成　　年度入学　学籍番号　　氏名

問題１

（1）　（2）　（3）

（4）　（5）　（6）

（7）　（8）　（9）

(10)　(11)　(12)

問題２

（1）　（2）　（3）

問題３（1）

(a)　(c)

(b)　(d)

（2）

（3－1）１：　２：　３：

（3－2）

S[0]	S[1]	S[2]	S[3]

問題４（1）

(a)　(c)

(b)　(d)

（2）

（3－1）１：　２：　３：

（3－2）

Q[0]	Q[1]	Q[2]	Q[3]

問題５（1）

(a)　(b)

(c)

（2）

（3）

問題６（1）

H	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
(a)												
(b)												
(c)												

（2）

（3）

（4）

問題７（1）

(a)　(b)

（2）

i	D[0]	D[1]	D[2]	D[3]	D[4]
初期状態	9	1	5	3	7
4					
3					
2					
1					

（3）

問題８（１）

(a)　　(b)

（２）

i	D[0]	D[1]	D[2]	D[3]	D[4]
初期状態	9	1	5	3	7
1					
2					
3					
4					

（３）

問題９（１）

(a)　　(b)

(c)　　(d)

(e)　　(f)

（２－１）

（２－２）

問題１０（１－１）

（１－２）

（１－３）

（２）(d)

(e)

(f)

(g)

問題１１

		重さの和の限界 w (kg)			
		0	1	2	3
部分問題	1	(0, ∅)		(30, {1})	
	2				
	3				

価格の総和を最大にする入れ方：