

Worksheet Complexity

Mike Fraser, Alex Stanhope

Week 8

In this worksheet you will be faced with problems to do with list and matrix algorithms, and answer questions regarding the complexity of the operations. Do not use a calculator anywhere.

1. There is an algorithm in speech recognition defined as follows:

$$\alpha_1(j) = \pi_j b_j(O_j) \text{ for } j = 1, \dots, N. \quad (1)$$

$$\alpha_t(j) = \left(\sum_{i=1}^N \alpha_{t-1}(i) a_{ij} \right) b_j(O_t) \text{ for } j = 1, \dots, N, t = 2, \dots, T \quad (2)$$

$$P(O|H) = \sum_{j=1}^N \alpha_T(j). \quad (3)$$

$P(O|H)$ is the probability that a set of observations O (ie spoken sounds) for a word can be produced by a given model H for a word. You don't need to know what all the notation means to be able to work out the complexity of this algorithm. All you need to know is that a_{ij} and $b_i(O_j)$ are matrices and π_j is a vector. (They are in fact probabilities associated with Markov Models and observation of sounds). The question is to determine the complexity of this algorithm in terms of the number of multiplications required.

2. Below is an algorithm that finds an element in a list.

```
List *findelement( char *s, List *x ) {
    if( x == NULL ) {
        return NULL ;
    }
    if( strcmp( x->contents, s ) == 0 ) {
        return x ;
    }
    return findelement( s, x->next ) ;
}
```

- (a) Given that the list has a length of n elements, what is the complexity of the algorithm?

(b) If the average length of a string is m elements, then what is the complexity expressed in both n and m ?

(c) Suppose we add a line

```
    if( strcmp( x->contents, s ) < 0 ) {  
        return NULL ;  
    }
```

before the final return statement, and we assume that the list is sorted, what is the complexity of this new function, expressed in just n ?

(d) Express the complexity in both m and n ?

(e) Has any of the *constant* factors changed for (c) and (d) ?

3. Below is an algorithm that multiplies a matrix of size $N \times M$ with a matrix of size $M \times K$ (resulting in a $N \times K$ matrix).

```
void multiply_matrix( double A[N][M], double B[M][K], double C[N][K] ) {  
    int i, j, k ;  
    for( i=0 ; i<N ; i++ ) {  
        for( k=0 ; k<K ; k++ ) {  
            double s = 0 ;  
            for( j=0 ; j<M ; j++ ) {  
                s = s + A[i][j] * B[j][k] ;  
            }  
            C[i][k] = s ;  
        }  
    }  
}
```

(a) What would you use to describe the size of the input (and hence to express the complexity in)?

(b) Derive the complexity of the matrix multiplication algorithm, if you count the multiplication operations?

(c) Derive the complexity of the matrix multiplication algorithm, if you count the addition operations?

(d) What is the complexity of the matrix multiplication algorithm, if K , N , and M are all equal to a number n ?

(e) Think of an algorithm to add a matrix of size $N \times M$ to another matrix of size $N \times M$. What is its complexity?

4. Consider the following program

```
typedef struct L {  
    struct L *bob ;  
    char element ;  
}
```

```

} List ;

List *i( char z, List *tail ) {
    List *new = calloc( 1, sizeof( List ) ) ;
    new -> element = z ;
    new -> bob = tail ;
    return new ;
}

List *makelist() {
    return i( 'h', i( 'e', i( 'l', i( 'l', i( 'o', NULL)))) ) ;
}

void reprint( List *x ) {
    while( x != NULL ) {
        printf( "%c", x->element ) ;
        x = x->bob ;
    }
}

void preprint( List *x ) {
    if( x != NULL ) {
        printf( "%c", x->element ) ;
        preprint( x->bob ) ;
    }
}

void postprint( List *x ) {
    if( x != NULL ) {
        postprint( x->bob ) ;
        printf( "%c", x->element ) ;
    }
}

void inprint( List *x ) {
    if( x != NULL ) {
        inprint( x->bob ) ;
        printf( "%c", x->element ) ;
        inprint( x->bob ) ;
    }
}

main() {
    List *y = makelist() ;
    reprint( y ) ;
    preprint( y ) ;
}

```

```
    postprint( y ) ;  
    inprint( y ) ;  
}
```

- (a) Draw the memory contents just before reprint is called. Assume that a List occupies two memory cells, the first one stores the pointer, the second one stores the character. Assume that calloc returns, when called, 12, 15, 27, 21, 18. Assume that y is stored in address 1.
- (b) What does the function reprint print? Assume x is stored in address 2. Draw the memory contents when going around the loop.
- (c) What does the function preprint print? Assume that x is stored on a stack, on positions 2, 3, 4, 5, ... Draw the memory contents at all stages.
- (d) What does the function postprint print?
- (e) What does the function inprint print?