

1

Quiz 1 Solution

Group A

1) What is the probability that three points on a circle will be on a semi-circle?

Solution: Denote the three points as point1, point2 and point3. Event A

i

is the following, two points lie in

the semi-circle that starts from pointi clockwise. P(A

i

) = 1 22

. Since point1, point2 and point3 are symmetric

and A

i

are mutually exclusive, thus the final probability is 3 × 1 22

. In general, if there are n points are the

probability they all lie in a semi-circle is n × 2n−1

1

.

2) An ant walks randomly on the edges of a cube. It starts from a vertex, and each step it has equal probability

to choose one of the three edges and walk to the other vertex of this edge. What is the expectation of the

number of steps for the ant to walk from one vertex to the opposite vertex.

Solution: Say the ant locates at (0,0,0). Let E

i

denote the expectation of steps needed to reach (1,1,1), if the

ant starts from a vertex with distance i to the (1,1,1). Use condition expectation, we can get the following,

E

3

=1+ E

2

,

E

2

=

1 3

(1 + E

3

) +

2 3

(1 + E

1

),

E

1

=

1 3

+

2 3

(1 + E

2

).

Then we can get E

3

= 10.

3) From a deck of 52 cards, you can pick one card each time without replacement. If the card color is black,

you win 1. If the card color is red, you lose 1. You can stop the game whenever you want. Questions: Will

you play the game? If you want, how much would you pay to play this game?

Solution: Let E(r, b) denote the expectation of the optimal strategy if r red cards and b black cards are left.

Then depend on the color of next card, we have the following.

E(r, b) = max(0,

r + r

b

(1 + E(r − 1,b)) +

r + b

b

(−1 + E(r, b − 1))).

E(26,26) can be computed recursively and its value is 2.62.

4) Given a coin with probability p of landing on heads after a flip, what is the probability that the number of

heads will ever equal the number of tails assuming an infinite number of flips?

Solution: Let’s only consider the case 0 <p< 1, otherwise the answer is trival. Denote state i as the number

of heads minus number of tails. Let state 0 be an absorbed state. Then the answer to the problem is equal to

the probability of the absorbing probability starting from 0. Denote P

i

as the absorbing probability starting

from i. Then we have

P

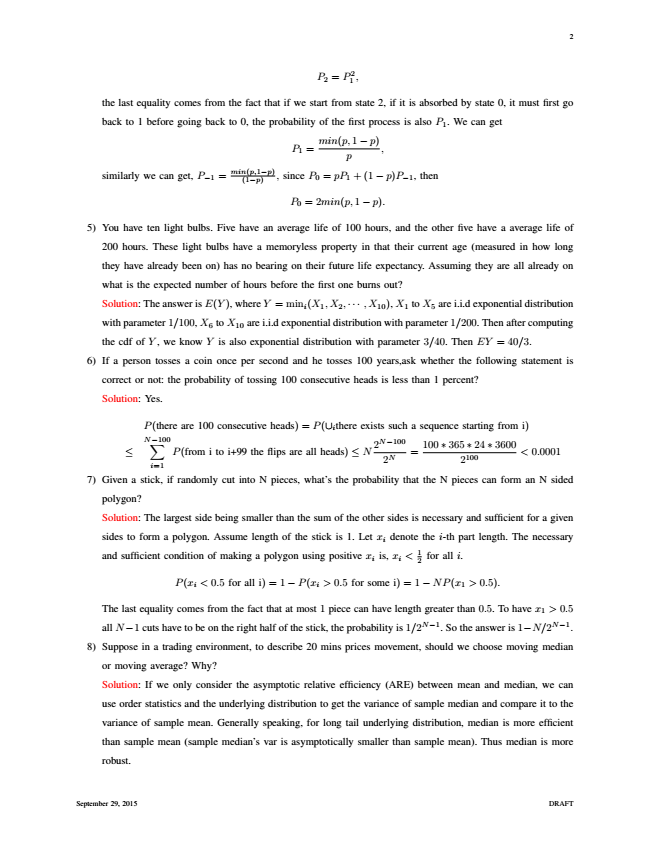
1

= (1 − p) + pP

2

,

September 29, 2015 DRAFT



2

P

2

= P 1 2

,

the last equality comes from the fact that if we start from state 2, if it is absorbed by state 0, it must first go

back to 1 before going back to 0, the probability of the first process is also P

1

. We can get

P

1

=

min(p,1 p

− p)

,

similarly we can get, P

−1

=

min(p,1−p) (1−p)

, since P

0

= pP

1

+ (1 − p)P

−1

, then

P

0

= 2min(p,1 − p).

5) You have ten light bulbs. Five have an average life of 100 hours, and the other five have a average life of

200 hours. These light bulbs have a memoryless property in that their current age (measured in how long

they have already been on) has no bearing on their future life expectancy. Assuming they are all already on

what is the expected number of hours before the first one burns out?

Solution: The answer is E(Y ), where Y = min

i

(X

1

,X

2

,··· ,X

10

), X

1

to X

5

are i.i.d exponential distribution

with parameter 1/100, X

6

to X

10

are i.i.d exponential distribution with parameter 1/200. Then after computing

the cdf of Y , we know Y is also exponential distribution with parameter 3/40. Then EY = 40/3.

6) If a person tosses a coin once per second and he tosses 100 years,ask whether the following statement is

correct or not: the probability of tossing 100 consecutive heads is less than 1 percent?

Solution: Yes.

P(there are 100 consecutive heads) = P(∪

i

there exists such a sequence starting from i)

≤

N−100∑

i=1

P(from i to i+99 the flips are all heads) ≤ N

2N−100 2N

=

100 ∗ 365 2100

∗ 24 ∗ 3600

< 0.0001

7) Given a stick, if randomly cut into N pieces, what’s the probability that the N pieces can form an N sided

polygon?

Solution: The largest side being smaller than the sum of the other sides is necessary and sufficient for a given

sides to form a polygon. Assume length of the stick is 1. Let x

i

denote the i-th part length. The necessary

and sufficient condition of making a polygon using positive x

i

is, x

i

< 1 2

for all i.

P(x

i

< 0.5 for all i)=1 − P(x

i

> 0.5 for some i)=1 − NP(x

1

> 0.5).

The last equality comes from the fact that at most 1 piece can have length greater than 0.5. To have x

1

> 0.5

all N−1 cuts have to be on the right half of the stick, the probability is 1/2N−1. So the answer is 1−N/2N−1.

8) Suppose in a trading environment, to describe 20 mins prices movement, should we choose moving median

or moving average? Why?

Solution: If we only consider the asymptotic relative efficiency (ARE) between mean and median, we can

use order statistics and the underlying distribution to get the variance of sample median and compare it to the

variance of sample mean. Generally speaking, for long tail underlying distribution, median is more efficient

than sample mean (sample median’s var is asymptotically smaller than sample mean). Thus median is more

robust.

September 29, 2015 DRAFT



3

However, whether median is better depends on the length of the interval of interest. Short period trend(SPT)

and long period trend(LPT) are different. The trade-off between these two is as follow: SPT should be more

sensitive and should capture signal ”outliers”. SPT is vulnerable to error but it can detect the change faster;

LPT should be more robust to outliers and reflect the overall trend during the time interval. 20 minutes may

be regarded as a LPT, so we prefer median. Note that in some situations, 20 minutes might be considered as

a SPT.

9) What is the average number of local maxima of a permutation of 1, ..., n, over all permutations? Maxima at

ends also count.

Solution: E ∑

i

I

i

Let X

i denote the value at location =

∑

i

EI

i

. EI

1

= 0.5 and EI

n

i. I

i

is the indicator function of location i being a maxima, then

= 0.5 since the maxima at ends count and I

0

= 1 if X

1

> X

2

which

has probability 1/2 due to symmetry. For any middle location m, it is a (X

m−1

,X

m

,X

m+1

) which has probability 1/3 due to the symmetry. So maxima E

∑

i

I

i

iff X

m

is the maximum in

= 1/2+ 1 3

(n−2) + 1/2 =

(n + 1)/3.

10) Give a one-line C expression to test whether a number is a power of 2.

Solution: x&(x − 1) == 0.

11) Implement a smartpointer in C++.

1

// SharedPtr.h

2

<template T>

3

4

class SharedPtr {

5

public :

6

// constructor

7

SharedPtr( const T &obj) :

8

p( new T(obj)), refCount( new std : : size\_t(1) ) {}

9

// copy constructor

10

SharedPtr( const SharedPtr &p\_) :

11

p(p\_) , refCount(p\_ . refCount) { ++

\*

refCount }

12

// copy assignment operator

13

SharedPtr& operator= (const SharedPtr&) ;

14

// Destructor

15

̃SharedPtr () ;

16

private :

17

T

\*

p;

18

std : : size\_t

\*

refCount ; // reference count

19

}

20

21

// Destructor definition

22

SharedPtr :: ̃ SharedPtr() {

23

// decrement the current reference count

24

// if it is 0, free the allocated resource

25

// if not, do nothing

26

if(−−

\*

refCount == 0) {

September 29, 2015 DRAFT



4

27

delete p;

28

delete refCount;

29

}

30

}

31

32

// copy assignment operator definition

33

SharedPtr& SharedPtr :: operator =( const SharedPtr &rhs) {

34

/ / increment the rhs pointer ' s reference count

35

++

\*

rhs . refCount;

36

/ / decrement the lhs pointer ' s reference coount

37

// and check whether it is 0 or not

38

// if it is 0, free the allocated resource

39

if ( −−

\*

refCount == 0) {

40

delete p;

41

delete refCount;

42

}

43

// copy rhs pointer and reference count to lhs

44

p = rhs . p ;

45

refCount = rhs . refCount;

46

return

\*

this ;

47

}

12) Reverse a linked list from position m to n. Do it in-place and in one-pass.

1

class Solution {

2

public :

3

ListNode

\*

reverseBetween(ListNode

\*

head , int m , int n) {

4

if (m>=n | | head==0 | | head−>next ==0 ) return head;

5

ListNode dummy=ListNode(0) ;

6

dummy . next=head;

7

ListNode

\*

mMinus=&dummy ,

\*

post ;

8

for(int i=m; −−i>0;)

9

mMinus=mMinus−>next;

10

ListNode

\*

pre=mMinus−>next,

\*

p=pre−>next;

11

while (n−m++>0){

12

post=p−>next;

13

p−>next=pre;

14

pre=p;

15

p=post;

16

}

17

mMinus−>next−>next=p;

18

mMinus−>next=pre ;

19

return dummy. next ;

20

}

21

};

13) Implement a program to find out whether there exist M days within the last N(N>=M) trading days that

September 29, 2015 DRAFT



5

the average closing price of these M days is at most P. Assume we have collected the history of the closing

prices of the last N trading days for a stock. Requirements: Inputs are positive integer M and N, M<= N;

An array of N float elements containing the closing prices of the last N trading days; And a float P. Please

design and implement the program in C, C++, Java or Python to produce the answer in most time and space

efficient way.

Solution: The program returns true iff MP is greater than the sum of the lowest M prices. Among many

methods min-heap method has the best performance, it has O(MN) worst case time complexity and O(1)

space complexity. For comparison, methods of using priority queue and std::sort are also included.

1

// Solution 1: Use priority queue.

2

bool ave\_lower\_P\_1( int N , int M , float P , vector<float > price) {

3

4

// imput the data into a priority queue

5

priority\_queue<float > price\_new;

6

for ( int i = 0; i < N ; i++) {

7

price\_new . push(−price[i ]) ;

8

}

9

10

// calculate the average of the lowest M days closing price

11

float sum = 0;

12

for ( int i = 0; i < M ; i++) {

13

sum += (−price\_new . top ());

14

price\_new . pop () ;

15

}

16

float ave = sum / M;

17

18

cout << ave << endl;

19

// return true or false

20

return (ave <= P) ;

21

}

22

// Solution 2: Use sort in c++.

23

bool ave\_lower\_P\_2( int N , int M , float P , vector<float > price) {

24

25

sort(price . begin () , price . end ());

26

27

// calculate the average of the lowest M days closing price

28

float sum = 0;

29

for ( int i = 0; i < M ; i++) {

30

sum += (price[i ]) ;

31

}

32

float ave = sum / M;

33

34

cout << ave << endl;

35

// return true or false

36

return (ave <= P) ;

September 29, 2015 DRAFT



6

37

}

38

// Solution 3: Use min heap in c++:

39

bool ave\_lower\_P\_3( int N , int M , float P , vector<float > price) {

40

41

make\_heap(price . begin () , price. end () , std : : greater<int >() ) ;

42

43

float sum = 0;

44

for ( int i = 0; i < M ; i++) {

45

sum += price . front () ;

46

cout << price . front () ;

47

pop\_heap(price. begin () , price. end () − 1 − i, std : : greater<int >() ) ;

48

price . pop\_back () ;

49

}

50

51

float ave = sum / M;

52

53

cout << ave << endl;

54

// return true or false

55

return (ave <= P) ;

56

}

14) Implement a string indexOf method that returns index of matching string.

1

#include <iostream>

2

#include <string >

3

#include <vector>

4

#include <ctime>

5

using namespace std;

6

7

class strMatch {

8

private :

9

vector<int > next;

10

void GetNext( const string& str) ;

11

public :

12

strMatch () {};

13

̃strMatch () {};

14

15

bool strStr( const string& haystack , const string& needle) ;

16

bool strStrKMP( const string& haystack , const string& needle) ;

17

};

18

19

// Brute force: time O(m

\*

n), space O(1)

20

bool strMatch : : strStr( const string& haystack , const string& needle) {

21

if (needle . empty () ) {

22

return true ;

23

}

September 29, 2015 DRAFT



7

24

25

for ( int i = 0; i < haystack . size () ; i++) {

26

if (haystack[i] == needle [0]) {

27

bool match = true ;

28

for ( int j = 0; j < needle . size () ; j++) {

29

if (haystack[i+j] != needle[j ]) {

30

match = false ;

31

break ;

32

}

33

}

34

35

if (match) {

36

return true ;

37

}

38

}

39

}

40

41

return false ;

42

}

43

44

// KMP: time O(m + n), space O(n)

45

void strMatch : : GetNext( const string& str) {

46

next . push\_back(−1) ;

47

int i = −1;

48

int j = 0;

49

50

while (j < str. size () − 1) {

51

//str[i] − prefixstr[j] − suffix

52

if (i == −1 | | str[j] == str[i ]) {

53

i++;

54

j++;

55

56

if (str[j] != str[i]) {

57

next . push\_back(i) ;

58

} else {

59

next . push\_back(next[i ]) ;

60

}

61

62

} else {

63

i = next[i ] ;

64

}

65

}

66

67

return ;

68

}

69

70

bool strMatch : : strStrKMP( const string& haystack, const string& needle) {

71

GetNext(needle) ;

72

September 29, 2015 DRAFT



8

73

int i , j ;

74

int haystackLen = haystack. size () ;

75

int needleLen = needle . size() ;

76

77

for (i = 0, j = 0; i < haystackLen && j < needleLen; ) {

78

// currently , match!

79

if (j == −1 | | haystack[i] == needle[j ]) {

80

i++;

81

j++;

82

} else {

83

// currently , NOT match..

84

j = next[j ] ;

85

}

86

}

87

88

if (j == needle. size () ) {

89

return true ;

90

} else {

91

return false ;

92

}

93

}

94

95

int main( int argc , char const

\*

argv []) {

96

strMatch soln;

97

98

string haystack;

99

cout << ”Input haystack: ”;

100

getline(cin , haystack) ;

101

102

string needle;

103

cout << ”Input needle: ”;

104

getline(cin , needle) ;

105

106

clock\_t now = clock () ;

107

cout << ”Brute force: ” << soln. strStr(haystack , needle) << endl;

108

clock\_t after = clock () ;

109

cout << ”Brute force run−time : ” << (after − now) /

110

( double ) (CLOCKS\_PER\_SEC / 1000) << ” ms” << endl ;

111

112

now = clock () ;

113

cout << ”KMP: ” << soln . strStrKMP(haystack, needle) << endl ;

114

after = clock () ;

115

cout << ”KMP run−time : ” << (after − now) /

116

( double ) (CLOCKS\_PER\_SEC / 1000) << ” ms” << endl ;

117

118

return 0;

119

}

15) Write a function to calculate exp(x).

September 29, 2015 DRAFT



9

1

#include <iostream>

2

#include <limits >

3

#include <cmath>

4

using namespace std;

5

6

class expFunction {

7

private :

8

double my\_power( double x , int n) ;

9

public :

10

expFunction () {};

11

̃expFunction () {};

12

13

double my\_exp( double x) ;

14

};

15

16

double expFunction : : my\_power( double x , int n) {

17

if (n == 0) {

18

return 1.0;

19

}

20

21

if (n%2 == 0) {

22

return my\_power(x , n /2)

\*

my\_power(x , n/2) ;

23

} else {

24

return x

\*

my\_power(x , n/2)

\*

my\_power(x, n /2) ;

25

}

26

}

27

28

double expFunction : : my\_exp( double x)

29

{

30

if (x < 0) {

31

return 1.0/ my\_exp(−x) ;

32

}

33

34

// Round up x when x is large so that

35

// eˆx = 1 + x + ... + xˆn/n! + O(xˆn) converges faster.

36

int roundup = ceil(x) ;

37

double x\_modified = x /roundup;

38

39

double result = 1.0;

40

double TaylorExpansionTerm = x\_modified;

41

int n = 1;

42

while (TaylorExpansionTerm > numeric\_limits<double >::min () ) {

43

result += TaylorExpansionTerm ;

44

TaylorExpansionTerm

\*

= (x\_modified/++n) ;

45

}

46

September 29, 2015 DRAFT



10

47

return my\_power(result , roundup) ;

48

}

49

50

51

int main( int argc , char const

\*

argv []) {

52

expFunction soln;

53

54

double power;

55

cout << ”Input the power: ” << endl ;

56

cin >> power;

57

58

cout << ”eˆ” << power << ” = ” << soln . my\_exp(power) << endl ;

59

60

return 0;

61

}

16) Given streaming data, design an algorithm to get approximate median of all previous data, use constant

memory.

Solution: Ideas: 1.Successive Bins/2.Reservoir sampling/3.Max and Min heap

Successive Bins: http://www.stat.cmu.edu/ ryantibs/papers/median.pdf (paper by Robert Tibshirani’s son...)

Sampling method: http://www.tks.informatik.uni-frankfurt.de/data/events/deis10/downloads/10452.ZelkeMariano.Slides.pdf

Max and Min heap: Hold two heaps, one max heap for values less than current median, one min heap for

values large than current median. The size of the two heaps diff at most 1 by constantly change the current

median.

Suppose we already have a heap structure in C++, max and min heap method is as follows.

1

int get\_median( int new\_number , Heap& min\_heap, Heap& max\_heap) {

2

if (max\_heap . size () && new\_number < max\_heap. top () ) {

3

max\_heap . enqueue(new\_number) ;

4

if (max\_heap . size() > min\_heap. size () ) {

5

min\_heap . enqueue(max\_heap . dequeue ());

6

}

7

} else {

8

min\_heap . enqueue(new\_number) ;

9

if (min\_heap . size() > max\_heap. size () +1) {

10

max\_heap . enqueue(min\_heap . dequeue ());

11

}

12

}

13

14

if (max\_heap . size ( ) ==min\_heap . size () ) {

15

return (max\_heap. top () + min\_heap . top ())/2;

16

} else {

17

return min\_heap. top () ;

18

}

19

}

September 29, 2015 DRAFT