***1. Algorithm pseudo-code***

Algorithm: greedy and score rating.

Input: a *MultiCalendar (0)* of my calendar and my friend’s calendar, *start* day, *end* day, *duration.*

Output: a *TimeHeap (1)* of *TimeNode (2)* that the TimeNode has max score is on the top of heap.

Function name: calculateTime(Date object);

Pseudo-code:

current = start;

mHeap = new TimeHeap(current);

while (current < end)

{

d = new TimeNode(current, current + duration);

mHeap.push(d);

current = *nextTime(current, mMultiCalendar)(3);*

}

Need defining:

(0) MultiCalendar object

(1) TimeHeap object

(2) TimeNode object

(3) nextTime() function

***2. MultiCalendar object***

Var: public object allCalendar = [[{}, {}], [{}]] array of array of object event in a date

Constructor: new MultiCalendar(friend ID) construct this.allCalendar as all calendar event of my calendar and my friend’s calendar.

Usage: mCal is an implement of MultiCalendar

mCal.allCalendar[new Date(year, month, day)] return an array of event on day/month/year

***3. TimeNode object***

Var: start, end, score

Function: all basic getting, setting function (getEnd(), setEnd(object Date), etc.)

Constructor: TimeNode(start, end); auto rate the time and save to score.

Usage: a TimeNode implement show that how many *score* is an event from start to end rated? It is used in TimeHeap.

***4. TimeHeap object***

Data structure: priority queue (max on the top)

Var: date, timeList (array of TimeNode), length, cache

Function: push (+ upHeap), pop (+ downHeap), getTop, backUp and all basic function

Constructor: TimeHeap(mDate); auto construct the heap with this.date is Monday of the week has mDate

Usage: used to save all available event times.

***5. nextTime(current, mMultiCalendar) function***

Input: a Date object and a MultiCalendar

Output: the next valid time