Rezervacije stolova u restoranu

# 1.Opis problemaž

Rezervacija stolova u restoranu problem je sa kojim se susreću mnogi ljudi. Neki samo u ulozi mušterije ali neki i u ulozi menadžera restorana. Menadžerima je naravno u interesu da restoran što bolje posluje pa samim tim da je popunjenost restorana što veća. Kako bi to bilo moguće potrebno je pažljivo osmotriti problem rezervacije stolova u restoranu. Treba voditi računa o tome kako popunjavamo raspored u danu ali i znati da se raspored može menjati kako bismo smestili nekoga za koga na prvi pogled nema mesta. Pri izmenama u rasporedu jako je važno voditi račna o tome da ne smemo neku rezervaciju otkazati ili promeniti joj vreme početka ili trajanje. Ovo naizgled ne deluje kao težak posao jer nama kao ljudima ovo može biti prirodno. Međutim kada je u pitanju računar koji nema subjektivni osećaj za ovakve stvari mora postojati algoritam po kome će on znati šta i kako je najbolje da radi. Na primerima sa manjim brojem stolova može delovati kao da je čovek dorastao zadatku i da je potreba za računarskim rešenjem neosnovana, međutim šta se desi kada broj stolova znatno poraste? Čovek više ne uspeva da isprati sve rezervacije i pokušaj rezervisanja može znatno da se oduži. Ovaj problem rešava računarski pristup. Naš pristup računarskom rešenju ima cilj da sprovede rezervisanje ali i da obezbedi da ni jedna rezervacija koja na prvi pogled ne može biti prihvaćena ne bude odbijena bez provere da li je to zaista jedino rešenje.

# 2. Pristup problemu

Sama rezervacija kada je ona moguća kao takva sa trenutnim stanjem rezervisanih stolova ne pravi problem i treba da bude automatski prihvaćena. Međutim kada rezervacija na prvi pogled nije moguća, odnosno kada je ne možemo prihvatiti bez izmene trenutnog stanja rasporeda ona ne sme biti prihvaćena dok se ne proveri da li je moguće izvršiti neku izmenu nad rasporedom pa je nakon izmeni prihvatiti.   
Samim tim se problem rezervacije stolova svodi na pokušavanje zakazivanja svih mogućih kombinacija termina gde će neka rešenja biti uspešna i prihvaćena dok će druga biti neuspešna zbog nezakazivanja svih termina te će u tom slučaju biti odbijena. Tek nakon i ove provere možemo biti sigurni da je rezervacija odbijena zbog nepostojanja nikakvog načina da bude prihvaćena. Tako težimo najvećoj popunjenosti restorana što nam je i bio cilj od samog početka.

Naš pristup dalje se svodi na izmenu rasporeda po principu ponovnog zakazivanja svih rezervacija iz rasporeda sa dodatkom nove rezervacije, tako da se prođu sve moguće kombinacije redosleda zakazivanja. Na taj način obezbeđuje se da je svaka moguća kombinacija isprobana i da će novi raspored imati pre svega sve stare rezervacije ali i novu koja pre izmena ne bi mogla postojati. Pri izmenama rasporeda vodi se računa o početku i trajnju, kao i broju gostiju u terminima koji su već bili zakazani, jedino što se sme menjati je broj stola za koji je zakazana rezervacija. Tako se rezervacije neće poremetiti i biće validne.   
  
3.Opis računarskog rešenja

Najvažnije funkcije korišćene u rešenju su scheduleTable, permutation i scramble.  
3.1. scheduleTable  
Funkcija scheduleTable je zadužena za zakazivanje termina za prvi slobodan sto, shodno broju gostiju.   
Parametri na osnovu kojih izvršava zakazivanje su: broj gostiju, vreme početka i dužina termina, ime za koje se vrši rezervacija.

3.2. permutation   
Funkcija permutation zadužena je za pravljenje permutacija redosleda po kom će se zakazivati rezervacije. Ona kao parametar prima početan redosled kako bi od njega napravila sve moguće permutacije.

3.3. scramble  
Funkcija scramble poziva se ukoliko je redovna rezervacija (scheduleTable) neuspešna. Ona je zadužena da pokuša sve moguće permutacije redosleda zakazivanja uz pomoć funkcije permutation koja joj govori koje su to permutacije koje treba da isproba. Ukoliko se i scramble završi neusešno korisniku se daje do znanja da rezervacija nije moguća. U slučaju da scramble uspe da pronađe kojim rasporedom može da zakaže sve rezervacije tako da se može prihvatiti nova rezervacija ona i biva prihvaćena.

4. Budućnost algoritma  
  
Opisani algoritam, nad kojim program radi, nije savršen. Pre svega složenost algoritma je O(n!), ali je dolazak do rešenja, ukoliko ono postoji, siguran. Program ima potencijal da se unapredi da radi nad bilo kojim brojem stolova.   
Vreme izvršavanja algoritma moglo bi se značajno poboljšati ukoliko bi se uvelo korišćenje threadova.