CS570
Analysis of Algorithms
Fall 2011
Final Exam

Name: ______________________
Student ID: __________________

____12:30 PM Section   ____2:00 PM Section

	Maximum	Received
Problem 1	20	
Problem 2	20	
Problem 3	15	
Problem 4	15	
Problem 5	15	
Problem 6	15	
Total	100	

2 hr exam
Close book and notes

If a description to an algorithm is required please limit your description to within 200 words, anything beyond 200 words will not be considered.

1) 20 pts
Mark the following statements as **TRUE**, **FALSE**. No need to provide any justification.
2)

3) **[TRUE/FALSE]** A minimum weight edge in a graph G must be in all minimum spanning trees of G.

4) **[TRUE/FALSE]** For a given problem with input size n, there must be some N that when n>N, a $\Theta(n\log n)$ algorithm runs faster than a $\Theta(n^2)$ algorithm.

5)

6) **[TRUE/FALSE]** If two minimum spanning trees on the same graph only have 2 edges in common, then those two edges must be the lowest costs edges in the graph.

7)

8) **[TRUE/FALSE]** If there is a polynomial time algorithm for some problem in NP, then all problems in NP can be solved in polynomial time.

9)

10) **[TRUE/FALSE]** If $A \le_p B \le_p C$ and $C \in NP$, then $A \in NP$.

11)

12)

13) **[TRUE/FALSE]** If $A \le_p B$ and $B \in$ NP - Complete, then $A \in$ NP - Complete.

14)

15) For the next four T/F questions, we define a *most vital arc* of a network as an arc whose deletion causes the largest decrease in the maximum *s-t*-flow value. Let f be an arbitrary maximum *s-t*-flow.

16) **[TRUE/FALSE]** A most vital arc is an arc *e* with the maximum value of *c(e)*.

17)

18) **[TRUE/FALSE]** A most vital arc is an arc *e* with the maximum value of *f(e)*.

19)

20) **[TRUE/FALSE]** A most vital arc is an arc *e* with the maximum value of *f(e)* among arcs belonging to some minimum cut.

21)

22) **[TRUE/FALSE]** An arc that does not belong to some minimum cut cannot be a most vital arc.

23)

24)

25)

26)

27)

28)

29)

30)

31)

32) 20 pts
Given an m X n integer matrix A and an m X 1 integer vector b, the 0-1 integer programming problem asks whether there is an n X 1 integer vector x with elements in the set {0,1} such that $Ax \leq b$. Prove that 0-1 integer programming problem is NP-complete.

33)

34)

35)

36)

37)

38)

39)

40)

41)

42)

43)

44)

45)

46)

47)

48)

49)

50)

51)

52)

53)

54)

55)

56)

57)

58)

59)

60)

61)

62)

63)

64)

65)

66)

67)

68)

69)
70)
71) 15 pts

There are four types of blood, A, B, AB, and O. Patients with blood type A can receive only blood types A or O in a transfusion, B can receive B or O, patients with type O can receive only O, and patients with AB can receive any of the four. Let s_O, s_A, s_B, and s_{AB} denote the supply in whole units of the different blood types on hand. Assume the hospital knows the projected demand for each blood type d_O, d_A, d_B, and d_{AB} for the coming week. Give an algorithm to help the hospital determine if the blood on hand would suffice for the projected need. Justify your algorithm and analyze the running time.

72)

73)

74)

75)

76)

77)

78)

79)

80)

81)

82)

83)

84)

85)

86)

87)

88)

89)

90)

91)

92)

93)

94)

95)

96)

97)

98)

99)

100)

101)

102)

103)

104)

105)

106)

107)

108) 15 pts

109) You are going on a long trip. You start on the road at mile post *0*. Along the way there are *n* hotels, at mile posts $a_1 < a_2 < \dots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance a_n), which is your destination. You'd ideally like to travel *200* miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel *x* miles during a day, the penalty for that day is $(200 - x)^2$. You want to plan your trip so as to minimize the total penalty (the sum, over all travel days, of the daily penalties). Give an efficient algorithm that determines the minimum total penalty and analyze the running time.

110)

111)

112)

113)

114)

115)

116)

117)

118)

119)

120)

121)

122)

123)

124)

125)

126)

127)

128)

129)

130)

131)

132)

133)

134)

135)

136)

137)

138)

139) 15 pts

A Max-Cut of an undirected graph G = (V,E) is defined as a cut C_{max} such that the number of edges crossing C_{max} is the maximum possible among all cuts. Consider the following algorithm.

140) (i) Start with an arbitrary cut C.

141) (ii) While there exists a vertex v such that moving v from one side of C to the other increases the number of edges crossing C, move v and update C.

142) (a) Does the algorithm terminate in time polynomial in |V|?

143)
144) (b) Prove that the algorithm is a 2 approximation, that is the number of edges crossing C_{max} is at most twice the number crossing C.
145)

146) 15 pts

Here is a divide-and-conquer algorithm that aims at finding a minimum spanning tree. Given a graph $G = (V, E)$, partition the set V of vertices into two sets $V1$ and $V2$ such that $|V1|$ and $|V2|$ differ by at most 1. Let $E1$ be the set of edges that are incident only on vertices in $V1$, and let $E2$ be the set of edges that are incident only on vertices in $V2$. Recursively solve a minimum spanning tree problem on each of the two subgraphs $G1 = (V1, E1)$ and $G2 = (V2, E2)$. Finally, select the minimum-weight edge in E that crosses the cut $(V1, V2)$, and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

147) Either argue that the algorithm correctly computes a minimum spanning tree of G, or provide an example for which the algorithm fails.

148)

149)

150)

151)

152)

153)

154)

155)

156)

157)

158)

159)

160)

161)

162)

163)

164)

165)

166)

167)

168)

169)

170)

171)

172)

173)

174)

175)

176)

177)
178) Additional Space

179)
180)
181)
182)
183)
184)
185)
186)
187)
188)
189)
190)
191)
192)
193)
194)
195)
196)
197)
198)
199)
200)
201)
202)
203)
204)

205)
206)
207)
208)
209)
210)
211)
212)
213)
214)
215)
216)
217)
218)
219)
220)
221)
222)
223)
224) Additional Space
225)