ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος
«Τεχνητή νοημοσύνη και έμπειρα συστήματα»

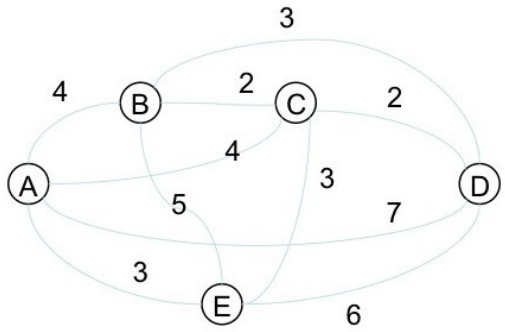
1η προαιρετική εργασία	***Πρόβλημα του πλανόδιου πωλητή***
Όνομα φοιτητή – Αρ. Μητρώου	Μιχαήλ Στυλιανίδης - Π19165
Ημερομηνία παράδοσης	13/06/2022

Εκφώνηση της άσκησης

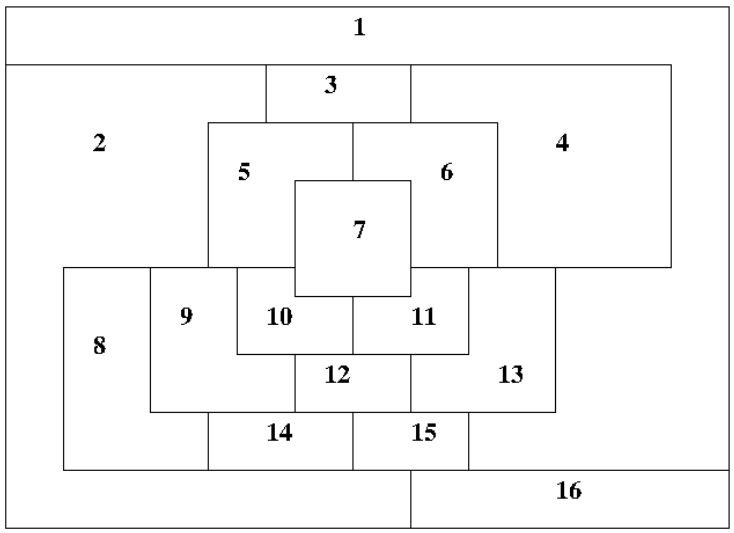
Θέμα προαιρετικής εργασίας για το μάθημα «Τεχνητή Νοημοσύνη και Έμπειρα Συστήματα».

<u>Η εργασία είναι ατομική. Επιλέξτε ένα από τα παρακάτω θέματα:</u>

Α. Αναπτύξτε πρόγραμμα επίλυσης του Traveling Salesman Problem με χρήση γενετικών αλγορίθμων και γλώσσα προγραμματισμού της επιλογής σας. Ο γράφος αποτελείται από πλήρως διασυνδεδεμένες πόλεις όπως φαίνεται στο παρακάτω σχήμα.



Β. Αναπτύξτε πρόγραμμα χρωματισμού του παρακάτω γράφου με χρήση γενετικών αλγορίθμων και γλώσσα προγραμματισμού της επιλογής σας. Τα διαθέσιμα χρώματα είναι 4: μπλε, κόκκινο, πράσινο, κίτρινο.



Χρησιμοποιείστε τυχαίο αρχικό πληθυσμό με πλήθος της δικής σας επιλογής. Χρησιμοποιείστε συνάρτηση καταλληλότητας και διαδικασία επιλογής γονέων σας της δικής σας επιλογής, επίσης. Χρησιμοποιείστε αναπαραγωγή με διασταύρωση ενός σημείου. Επιλέξτε αν θέλετε να κάνετε και μερική ανανέωση πληθυσμού σε κάποιο ποσοστό π.χ. 30% και μετάλλαξη ενός ψηφίου π.χ. στο 10% του πληθυσμού. Παραδοτέα της εργασίας είναι μία σύντομη αναφορά που να περιλαμβάνει τον τρόπο δράσης του υπολογιστή σύμφωνα με τον αλγόριθμο επίλυσης και παραδείγματα εκτέλεσης του προγράμματος που αναπτύξατε.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Table of Contents

1 Γενική Περιγραφή της λύσης

Το πρόβλημα που επιλύεται είναι το traveling salesman problem. Η εργασία έχει υλοποιηθεί σε console application με γλώσσα προγραμματισμού C#. Η εφαρμογή υλοποιεί για το traveling salesman problem τη γενική μορφή του γενετικού αλγορίθμου, η οποία έχει απεικονιστεί ως ένα interface με τις εξής μεθόδους:

- **Solve()**: Συγκεντρώνει τα βήματα των υπόλοιπων μεθόδων για τη λύση του προβλήματος
- **List<TPopulation> CreatePopulation()**: Δημιουργεί τον αρχικό πληθυσμό ως μία λίστα του πληθυσμού του εκάστοτε προβλήματος
- **List<TSolution> CreateRandomSolutions(List<TPopulation> population)**: Δημιουργεί μία λίστα από τυχαίες λύσεις με βάση τον αρχικό πληθυσμό
- **void RateSolutions(List<TSolution> solutions)**: Αξιολογεί τις λύσεις με βάση μία συνάρτηση καταλληλότητας
- **List<TCouple> SelectParents(List<TSolution> paths)**; Επιλέγει γονείς από τις διαθέσιμες λύσεις
- **List<TSolution> BreedNewPopulation(List<TCouple> parents)**: Δημιουργεί απογόνους μέσω των επιλεγμένων γονέων
- **bool AreSolutionsConverging(List<TSolution> solutions):** Ελέγχει εάν οι λύσεις συγκλίνουν

Στο συγκεκριμένο πρόβλημα το TPopulation αφορά τις πόλεις (City) το TSolution τις διαδρομές (Path) και το TCouple τα ζευγάρια διαδρομών που συμβολίζουν τους γονείς (PathPair).

Ακολουθεί η αναλυτική περιγραφή του προγράμματος

2 Δημιουργία πληθυσμού

Ο αρχικός πληθυσμός αποτελείται πάντα από ένα συγκεκριμένο αριθμό πόλεων (ο οποίος καθορίζεται στην αρχή του προγράμματος). Οι πόλεις αναπαριστώνται δυαδικά από το 0 έως αυτό τον αριθμό. Στην αρχή του προγράμματος δημιουργείται μία λίστα όπου αποθηκεύονται όλες οι αρχικές πόλεις. Για παράδειγμα εάν ο αρχικός πληθυσμός ήταν μήκους 10 η λίστα έχει ως εξής:

```
Starting cities:
Cities:
-0000-0001-0010-0011-0100-0101-0110-0111-1000-1001-
```

Για τον καθορισμό του αριθμού των πόλεων χρησιμοποιείται η ρυθμιστική κλάση TspOptions. Εκεί περιλαμβάνονται όλες οι παράμετροι του συστήματος. Συγκεκριμένα, πέρα από τον αριθμό των πόλεων ορίζεται ο αριθμός των τυχαίων λύσεων που θα παραχθούν, το ελάχιστο και το μέγιστο κόστος που μπορεί να έχει μία διαδρομή σε σχέση με κάποια άλλη και ο τρόπος με τον οποίο αναπαριστώνται τα δεδομένα, δηλαδή σε δυαδική ή δεκαδική μορφή.

3 Παραγωγή τυχαίων λύσεων

Για την παραγωγή τυχαίων λύσεων χρησιμοποιείται η κλάση PathManagement. Συγκεκριμένα, περιλαμβάνει μέθοδο που δημιουργεί μία λίστα από τυχαίες διαδρομές (Path class). Η διαδρομή περιλαμβάνει μία λίστα με τις διαδοχικές πόλεις που περιέχει, το κόστος της διαδρομής (το οποίο υπολογίζεται σε μεταγενέστερο στάδιο), την απόσταση μεταξύ της κάθε πόλης (το οποίο ορίζεται τυχαία σε κάθε εκτέλεση με βάση τα όρια) με όλες τις υπόλοιπες καθώς και άλλες πληροφορίες που θα αναλυθούν αργότερα. Για την τυχαιότητα των διαδρομών γίνεται χρήση της συνάρτησης Random της C# και μέσω αυτής ανακατεύονται οι πόλεις κρατώντας την αρχική και την τελική, οι οποίες ταυτίζονται, σταθερές. Για παράδειγμα, εάν ο πληθυσμός είναι μήκους 10 και ο αριθμός των τυχαίων λύσεων 15 μία περίπτωση τυχαίων λύσεων είναι η εξής:

```
Starting random paths
City Paths with costs:
-0000-0100-0111-0011-0010-0001-0110-0101-1000-1001-0000-
-0000-0001-1001-0111-0010-0100-0101-0110-0011-1000-0000-
-0000-0010-0011-0110-1000-0100-0101-0001-1001-0111-0000-
-0000-0100-0011-0111-0001-0101-0010-1001-0110-1000-0000-
-0000-0101-0100-0110-0011-1001-0001-0111-0010-1000-0000-
-0000-0101-0001-1001-0100-0110-0111-0011-1000-0010-0000-
-0000-0001-0010-0011-0110-0100-1001-0101-1000-0111-0000-
-0000-1000-0100-0110-0011-0001-1001-0010-0101-0111-0000-
-0000-0100-0011-0111-1001-0101-1000-0001-0110-0010-0000-
-0000-0100-0101-0010-1001-1000-0111-0110-0011-0001-0000-
-0000-0011-0111-0100-0110-0001-1000-1001-0010-0101-0000-
-0000-0010-1000-0011-0001-0111-0101-0100-0110-1001-0000-
-0000-0011-0110-1000-1001-0111-0100-0010-0001-0101-0000-
-0000-0101-0001-0010-0110-0100-0111-0011-1000-1001-0000-
-0000-1001-0110-0101-0011-0010-0111-0001-1000-0100-0000-
```

4 Αξιολόγηση λύσεων

Για την αξιολόγηση των λύσεων έχει δημιουργηθεί η κλάση Cost. Εκεί υπολογίζεται το κόστος της κάθε διαδρομής με βάση την απόσταση μεταξύ της κάθε πόλης με όλες τις υπόλοιπες. Στην περίπτωση του traveling salesman problem όσο πιο μικρό είναι το κόστος τόσο καλύτερα. Όμως, στην τεχνική αξιολόγησης της ρουλέτας οι μεγαλύτερες τιμές θεωρούνται καλύτερες. Συνεπώς, με σκοπό την εφαρμογή αυτής της τεχνικής οι τιμές του κόστους αντιστρέφονται. Επιπλέον, κάθε φορά που υπολογίζεται το κόστος μιας διαδρομής συγκρίνεται με το καλύτερο έως τώρα και το αντικαθιστά εφόσον κριθεί ανώτερο. Αυτό συμβαίνει, διότι ο πληθυσμός δεν συγκλίνει πάντοτε στην καλύτερη λύση που έχει εξεταστεί. Ένα παράδειγμα υπολογισμού κόστους διαδρομών είναι το εξής:

```
Starting random paths
City Paths with costs:
-0000-0111-1000-1001-0100-0011-0110-0101-0010-0001-0000- -> 139
-0000-0011-0110-1001-0001-0010-1000-0101-0111-0100-0000- -> 148
-0000-0110-0011-0100-0001-0111-1001-1000-0010-0101-0000- -> 129
-0000-1001-0100-0110-0101-0001-0010-0111-1000-0011-0000- -> 136
-0000-0010-0110-0001-0011-0101-0100-0111-1001-1000-0000- -> 146
-0000-0110-1001-0001-0011-0010-1000-0100-0111-0101-0000- -> 143
-0000-1000-0100-1001-0011-0010-0111-0001-0101-0110-0000- -> 118
-0000-1001-0101-0100-0011-0001-0010-1000-0111-0110-0000- -> 140
-0000-0110-1001-0010-0001-0101-0011-0100-1000-0111-0000- -> 142
-0000-0011-0001-1001-0110-1000-0101-0100-0010-0111-0000- -> 175
-0000-0100-0011-0001-0110-0111-0101-0010-1001-1000-0000- -> 141
-0000-1000-0010-0111-1001-0101-0001-0011-0110-0100-0000- -> 138
-0000-0010-0001-1001-0111-0011-1000-0101-0100-0110-0000- -> 136
-0000-0011-0101-1000-0010-0110-0111-0100-0001-1001-0000- -> 141
-0000-0110-0010-0011-0111-1001-0100-1000-0101-0001-0000- -> 141
```

Σημείωση: Το κόστος όταν εμφανίζεται στην οθόνη αντιστρέφεται ξανά στην αρχική του μορφή, ώστε να διαβάζεται ευκολότερα από τον χρήστη. Οπότε, το μικρότερο κόστος θεωρείται καλύτερο κατά την οπτικοποίηση.

5 Επιλογή γονέων

Για την επιλογή γονέων χρησιμοποιείται η κλάση Breeding. Συγκεκριμένα περιέχει μέθοδο, η οποία δημιουργεί N/2 ζευγάρια γονέων, όπου N είναι το πλήθος των διαδρομών. Οι γονείς επιλέγονται τυχαία, αλλά δίνοντας μεγαλύτερη πιθανότητα στα μονοπάτια με το καλύτερο κόστος, με βάση την τεχνική της ρουλέτας. Ένα παράδειγμα παραγωγής γονέων είναι το εξής:

Από την παρακάτω λίστα διαδρομών:

```
City Paths with costs:
-0000-1001-0010-0100-0001-0101-1000-0011-0110-0111-0000- -> 169
-0000-0100-1001-1000-0010-0110-0111-0001-0101-0011-0000- -> 137
-0000-0101-0111-0011-0010-0110-1000-1001-0001-0100-0000- -> 141
-0000-0110-1001-0111-0010-1000-0100-0001-0011-0101-0000- -> 108
-0000-0011-0101-0110-0111-0010-1000-1001-0100-0001-0000- -> 96
-0000-0101-0010-0111-0011-1001-0110-0100-1000-0001-0000- -> 117
-0000-1001-0100-1000-0010-0101-0111-0001-0011-0110-0000- -> 132
-0000-0101-0011-1000-0010-0100-0001-0111-0110-1001-0000- -> 140
-0000-0001-0010-0111-1001-0011-0101-0110-0100-1000-0000- -> 139
-0000-0001-0101-1000-0110-0011-1001-0010-0111-0100-0000- -> 173
-0000-0101-0110-0100-0001-0111-0010-1000-0011-1001-0000- -> 128
-0000-0110-0100-0010-0111-0001-1001-0011-0101-1000-0000- -> 154
-0000-1000-0110-0011-0111-0100-0001-0101-0010-1001-0000- -> 148
-0000-0101-0011-0001-0110-0111-0100-1000-0010-1001-0000- -> 131
-0000-0001-0111-1001-0110-0010-1000-0011-0101-0100-0000- -> 159
```

Παράγονται οι εξής γονείς:

```
Selected parents
Couple 1
-0000-0101-0010-0111-0011-1001-0110-0100-1000-0001-0000- -> 117
-0000-1000-0110-0011-0111-0100-0001-0101-0010-1001-0000- -> 148
Couple 2
-0000-0101-0011-1000-0010-0100-0001-0111-0110-1001-0000- -> 140
-0000-0001-0010-0111-1001-0011-0101-0110-0100-1000-0000- -> 139
Couple 3
-0000-0011-0101-0110-0111-0010-1000-1001-0100-0001-0000- -> 96
-0000-0101-0110-0100-0001-0111-0010-1000-0011-1001-0000- -> 128
Couple 4
-0000-0101-0110-0100-0001-0111-0010-1000-0011-1001-0000- -> 128
-0000-0101-0011-0001-0110-0111-0100-1000-0010-1001-0000- -> 131
Couple 5
-0000-0101-0011-0001-0110-0111-0100-1000-0010-1001-0000- -> 131
-0000-0011-0101-0110-0111-0010-1000-1001-0100-0001-0000- -> 96
Couple 6
-0000-0110-1001-0111-0010-1000-0100-0001-0011-0101-0000- -> 108
-0000-1000-0110-0011-0111-0100-0001-0101-0010-1001-0000- -> 148
Couple 7
-0000-0101-0110-0100-0001-0111-0010-1000-0011-1001-0000- -> 128
-0000-0100-1001-1000-0010-0110-0111-0001-0101-0011-0000- -> 137
Couple 8
-0000-1001-0010-0100-0001-0101-1000-0011-0110-0111-0000- -> 169
-0000-0101-0011-0001-0110-0111-0100-1000-0010-1001-0000- -> 131
```

Σημείωση: Σε περίπτωση που ο αριθμός των μονοπατιών είναι μονός, τότε ο αριθμός των γονέων βγαίνει κατά 1 μεγαλύτερος από αυτόν των μονοπατιών. Στη συνέχεια, για να μην αλλοιωθεί το μέγεθος του αρχικού πληθυσμού το τελευταίο ζευγάρι παράγει 1 μόνο απόγονο

6 Γένεση απογόνων

Για τη δημιουργία απογόνων χρησιμοποιείται και πάλι η κλάση Breeding. Εκεί περιέχεται μέθοδος η οποία δέχεται μία λίστα από γονείς (List<PathPair>) και δημιουργεί ένα νέο πληθυσμό. Κάθε ζευγάρι γονέων γεννάει δύο απογόνους. Ο πρώτος αποτελείται σε γενικές γραμμές από το πρώτο μισό κομμάτι του ενός γονέα και το δεύτερο μισό του άλλου, ενώ ο δεύτερος απόγονος το αντίστροφο. Ωστόσο, αυτό δεν είναι απόλυτο. Κάποιες φορές με αυτή τη μέθοδο δημιουργούνται μη νόμιμες διαδρομές, που περιέχουν για παράδειγμα μία πόλη πολλές φορές. Για να αντιμετωπιστεί αυτό το πρόβλημα, πριν μπει μία πόλη στην κάθε διαδρομή ελέγχεται εάν υπάρχει ήδη, και αν ναι τότε αγνοείται. Εάν το δεύτερο μισό του δεύτερου γονέα φτάσει στο τέλος του και δεν έχει ολοκληρωθεί η διαδρομή, τότε η αναζήτηση χρωμοσωμάτων εξετάζει σειριακά και το πρώτο κομμάτι του δεύτερου γονέα έως ότου συμπληρωθούν όλες οι πόλεις. Παρακάτω φαίνεται ένα παράδειγμα γένεσης απογόνων:

Από τους εξής γονείς:

```
Couple 1
-0000-0101-1001-0111-0011-0001-0100-1000-0110-0010-0000- -> 126
-0000-0101-0001-0010-0011-1000-0100-0110-1001-0111-0000- -> 117
Couple 2
-0000-1001-1000-0001-0100-0011-0101-0110-0111-0010-0000- -> 154
-0000-0101-1001-0111-0011-0001-0100-1000-0110-0010-0000- -> 126
Couple 3
-0000-0110-0101-1001-0010-0100-1000-0111-0011-0001-0000- -> 132
-0000-1001-1000-0001-0100-0011-0101-0110-0111-0010-0000- -> 154
Couple 4
-0000-0011-0111-1000-1001-0101-0001-0100-0010-0110-0000- -> 154
-0000-0101-0001-0010-0011-1000-0100-0110-1001-0111-0000- -> 117
Couple 5
-0000-0100-0011-1000-0001-0110-0111-0101-0010-1001-0000- -> 155
-0000-0001-0100-0110-0011-1000-0111-0101-1001-0010-0000- -> 128
Couple 6
-0000-0011-1000-0010-0101-0001-1001-0100-0110-0111-0000- -> 135
-0000-0100-0011-0001-1000-0010-0111-1001-0110-0101-0000- -> 144
Couple 7
-0000-1001-1000-0001-0100-0011-0101-0110-0111-0010-0000- -> 154
-0000-0001-0100-0110-0011-1000-0111-0101-1001-0010-0000- -> 128
Couple 8
-0000-0001-0100-0110-0011-1000-0111-0101-1001-0010-0000- -> 128
-0000-0100-0011-0001-1000-0010-0111-1001-0110-0101-0000- -> 144
```

Γεννήθηκε ο παρακάτω νέος πληθυσμός:

```
New generation:
City Paths with costs:
-0000-0101-1001-0111-0011-1000-0100-0110-0001-0010-0000- -> 124
-0000-0101-0001-0010-0011-0100-1000-0110-1001-0111-0000- -> 114
-0000-1001-1000-0001-0100-0110-0010-0101-0111-0011-0000- -> 140
-0000-0101-1001-0111-0011-0110-0010-1000-0001-0100-0000- -> 141
-0000-0110-0101-1001-0010-0011-0111-1000-0001-0100-0000- -> 150
-0000-1001-1000-0001-0100-0111-0011-0110-0101-0010-0000- -> 149
-0000-0011-0111-1000-1001-0100-0110-0101-0001-0010-0000- -> 147
-0000-0101-0001-0010-0011-0100-0110-0111-1000-1001-0000- -> 138
-0000-0100-0011-1000-0001-0111-0101-1001-0010-0110-0000- -> 161
-0000-0001-0100-0110-0011-0111-0101-0010-1001-1000-0000- -> 130
-0000-0011-1000-0010-0101-0111-1001-0110-0100-0001-0000- -> 121
-0000-0100-0011-0001-1000-1001-0110-0111-0010-0101-0000- -> 157
-0000-1001-1000-0001-0100-0111-0101-0010-0110-0011-0000- -> 151
-0000-0001-0100-0110-0011-0101-0111-0010-1001-1000-0000- -> 136
-0000-0001-0100-0110-0011-0010-0111-1001-0101-1000-0000- -> 120
```

7 Εύρεση της καλύτερης λύσης

Μετά την πρώτη τυχαία παραγωγή λύσεων, ο αλγόριθμος δημιουργεί νέους πληθυσμούς και τους αξιολογεί έως ότου ο πληθυσμός αρχίζει και συγκλίνει σε μία λύση. Όταν συγκλίνει κατά 95% σε αυτή τη λύση τότε η εκτέλεση σταματάει και εμφανίζεται η καλύτερη διαδρομή που έχει βρεθεί μέχρι αυτή τη στιγμή. Ακολουθούν μερικά παραδείγματα εκτέλεσης του αλγορίθμου, με διάφορες παραμέτρους. Σε αυτά τα παραδείγματα εμφανίζονται αρχικά οι διαθέσιμες πόλεις, μετά οι πρώτες τυχαίες λύσεις μαζί με το κόστος τους και στο τέλος η καλύτερη διαδρομή που βρέθηκε μετά τη σύγκλιση του πληθυσμού.

Πλήθος πόλεων: 10

Πλήθος διαδρομών: 15

```
Starting cities:
Cities:
-0000-0001-0010-0011-0100-0101-0110-0111-1000-1001-
Starting random paths
City Paths with costs:
-0000-0001-1001-0011-0010-0101-1000-0111-0100-0110-0000- -> 121
-0000-1000-0111-1001-0110-0001-0011-0010-0100-0101-0000- -> 142
-0000-1001-0001-0101-1000-0111-0010-0100-0011-0110-0000- -> 106
-0000-0101-0011-1001-0111-0110-0100-0010-0001-1000-0000- -> 151
-0000-1000-0011-0110-0001-0010-0101-1001-0111-0100-0000- -> 161
-0000-0110-0011-0010-0111-0101-1000-1001-0001-0100-0000- -> 123
-0000-0001-1000-0111-0011-0101-1001-0100-0110-0010-0000- -> 151
-0000-0010-0111-0001-0101-0100-1001-1000-0011-0110-0000- -> 149
-0000-0111-1001-0011-0100-0010-0001-0110-1000-0101-0000- -> 128
-0000-0101-1001-0010-0001-0011-0100-0111-1000-0110-0000- -> 123
-0000-0100-0110-0111-1001-0001-0010-0011-0101-1000-0000- -> 140
-0000-0110-1001-0101-0111-0010-0100-0001-0011-1000-0000- -> 146
-0000-0110-1000-1001-0010-0011-0001-0101-0100-0111-0000- -> 120
-0000-0110-0010-0100-1000-0111-0101-0001-1001-0011-0000- -> 159
-0000-0011-1000-0010-0001-0100-0111-0110-0101-1001-0000- -> 170
The best path that was found is:
-0000-0110-1000-1001-0010-0100-0001-0101-0011-0111-0000-
With cost: 102
```

Πλήθος πόλεων: 20

Πλήθος διαδρομών: 15

```
Starting cities:
Cities:
-00000-00001-00010-00011-00100-00101-00110-00111-01000-01001-01010-01011-01100-01101-01110-01111-10000-10001-10010-10011-
Starting random paths
City Paths with costs:
-00000-00001-10011-01100-10000-01001-10001-00011-01110-00110-01101-10010-00111-00101-01111-01011-00010-01010-00100-01000-00000- -> 289
-00000-00001-10001-01001-01100-01011-00010-01111-01000-01110-10010-01010-00011-00100-00111-00101-01101-00110-10011-10000-00000- -> 278
-00000-01111-01101-10000-00110-01001-01000-01110-01011-01100-00101-00100-10011-01010-00010-10010-00111-10001-00011-00001-00000- -> 328
-00000-00011-00100-01000-00110-10001-10011-00010-01101-01110-01010-01011-00101-10010-01001-00001-10000-00111-01100-01111-00000- -> 286
-00000-10010-10000-01010-01011-00101-00110-01001-01101-01000-01111-10001-01100-00100-00011-00010-01110-10011-00111-00001-00000- -> 269
-00000-01001-01100-01011-01010-01111-00011-01000-10011-00100-00010-00111-10000-10001-00001-01101-01110-10010-00101-00110-00000- -> 273
-00000-01110-00111-00011-01101-00101-00010-10010-01010-00110-01011-01001-10000-01000-10011-01111-00100-00001-10001-01100-00000- -> 271
-00000-01100-00110-01010-10011-01011-00001-01000-01111-01001-00100-10001-01110-00011-00010-00101-10010-10000-01101-00111-00000- -> 299
-00000-10000-01000-10010-01010-00101-01011-01111-01110-01001-01101-10011-00111-00010-10001-00011-00100-00110-01100-00001-00000- -> 275
-00000-01000-10000-01011-10011-01101-01111-01100-10001-01110-00010-10010-01010-01001-00111-00011-00001-00100-00101-00110-00000- -> 289
-00000-00010-01100-00111-10000-00100-01001-00101-01000-00110-00011-10001-10011-01101-01110-01011-01010-00001-10010-01111-00000- -> 308
-00000-10001-00101-00111-01000-01100-01101-01110-00010-01011-00100-00011-00001-01010-10010-10000-01001-01111-10011-00110-00000- -> 285
-00000-10000-00011-10010-01010-10011-01111-01100-01000-00001-01011-00110-10001-00010-01101-01110-00111-01001-00100-00101-00000- -> 306
-00000-01101-00111-10010-01001-01011-10011-00001-00011-01111-00110-10000-00100-01000-00101-10001-01110-01010-01100-00010-00000- -> 302
-00000-01001-01111-01110-10000-01000-01100-01101-01011-00001-10011-00101-10001-10010-00110-00010-00100-00111-01010-00011-00000- -> 280
The best path that was found is:
-00000-10000-01000-10010-01010-00101-01011-01111-01110-01001-10011-00100-00001-10001-01100-00111-00011-01101-00010-00110-00000-
With cost: 247
```

Πλήθος πόλεων: 30

Πλήθος διαδρομών: 30

```
Starting cities:
Cities:
-00000-00001-00010-00011-00100-00101-00110-00111-01000-01001-01010-01011-01100-01101-01110-01111-10000-10001-10010-10011-10100-10101-10110-10111-11000-11001-11010-11011-11100-11101-
Starting random paths
City Paths with costs:
-00000-11000-11100-11011-01010-10001-10100-01000-10110-00110-10000-01110-11001-01011-11101-01001-00101-00100-00001-10010-00011-11010-10111-10011-01100-10101-00111-00010-01111-01101-00000- -> 416
-00000-00001-01100-01110-01101-10100-01111-10110-10001-11010-01011-00100-00011-01000-00111-00101-01010-11101-00010-10000-11011-10111-10101-00110-10010-01001-11100-11000-11001-10011-00000- -> 424
-00000-00011-10010-10110-10001-01101-01110-01111-11001-11000-11100-10100-01000-11011-01100-00110-11010-01010-10111-00101-10011-01011-00001-01001-00100-00111-10101-00010-11101-10000-00000- -> 403
-00000-01011-10010-10001-00011-01110-00110-01111-01000-11101-11010-01001-00100-11000-11100-01100-11001-01101-00001-00101-11011-00010-10101-10000-10110-10100-10111-01010-10011-00111-00000- -> 366
-00000-11100-11010-01000-11000-10101-00101-10001-11011-10100-10110-01100-10111-01011-01110-01111-00011-00010-01001-11101-10010-00001-00100-11001-01101-10000-10011-01010-00111-00110-00000- -> 387
-00000-00110-10111-00111-01010-11011-10100-10101-10110-11001-01100-00001-11100-00010-01000-10011-11010-01110-01101-10000-00100-11101-01111-11000-00011-00101-10001-10010-01001-01011-00000- -> 396
-00000-00001-10000-11000-00010-01101-01001-10011-00101-11011-01111-01110-11100-11010-11001-00011-10100-01100-00111-10111-10001-01000-00100-10101-01011-11101-10110-01010-00110-10010-00000- -> 457
-00000-01100-11100-01101-11011-01010-01011-10001-00111-10011-01111-10000-11101-10101-00001-01001-11010-10100-10010-01000-00010-00101-00011-11000-11001-10110-00110-00100-10111-01110-00000- -> 361
-00000-10010-10111-01001-00110-00011-11010-10011-00001-10110-11001-01100-01010-01101-11000-10101-10100-11100-10000-00010-11011-00111-11101-00101-10001-00100-01111-01110-01011-01000-00000- -> 396
-00000-00110-11011-11010-11100-01101-01110-01100-10011-00001-10100-00111-01000-00010-01010-10111-11101-00011-10010-10000-10001-11000-10110-01011-01001-00100-11001-00101-10101-01111-00000- -> 455
-00000-11100-00001-01110-01010-10000-10011-10110-01101-01111-11010-10111-01011-00010-00111-10010-10101-00100-01000-01100-10100-10001-11101-11000-00011-00101-11001-00110-01001-11011-00000- -> 436
-00000-00111-11101-11000-01010-00100-10001-00101-11001-00011-10010-01011-10011-10100-10101-01101-00010-11011-00110-10110-01000-01001-00001-01100-11010-01111-10000-01110-10111-11100-00000- -> 434
-00000-00101-00110-00010-10101-10001-10010-00001-00100-11000-00111-01100-01001-11010-10110-11001-01101-01110-10100-00011-11100-10111-01010-10011-01000-11101-01011-11011-10000-01111-00000- -> 380
-00000-10010-01110-10001-01101-10100-00111-00001-10000-11010-11011-00011-00110-11100-01000-00101-11000-01010-10110-10011-00010-01111-11101-01011-10111-01001-00100-11001-10101-01100-00000- -> 450
-00000-01010-10101-00110-01000-00010-01101-11100-01110-00111-10100-10110-11101-11000-11001-11010-00001-10010-00101-01011-10000-10111-00100-01001-01100-00011-01111-10011-11011-10001-00000- -> 429
-00000-10000-00010-11001-11011-01010-00111-10100-00101-10010-10011-00110-00011-10110-11101-01110-01111-10101-01101-00001-01000-01011-11010-10111-01001-11000-10001-00100-11100-01100-00000- -> 382
-00000-10110-11101-11001-11011-11000-01100-11100-01111-10100-01011-00010-00011-11010-10000-10010-10001-00100-00110-00111-00001-01000-01001-01101-10101-01010-10011-01110-00101-10111-00000- -> 434
-00000-01111-01000-10000-01011-11011-00010-00011-11101-10101-10001-00101-11100-00001-11001-01101-10010-01100-00110-11000-10110-10111-11010-00100-01010-10100-10011-01110-01001-00000- -> 446
-00000-10011-01110-00110-01010-00101-01111-10001-11000-10110-00111-01100-10000-10111-00001-10101-00011-01000-01101-00100-11010-00010-11011-01001-11101-11001-11100-01011-10100-10010-00000- -> 452
-00000-10000-10011-11100-11000-10110-00101-01000-10010-10101-00011-11010-00110-10111-11001-00010-11101-11011-01100-01101-01001-10001-00111-01011-00100-10100-01110-01111-01010-00001-00000- -> 429
-00000-11000-01101-11010-01000-10000-10011-10111-01100-10001-00010-11001-00001-01110-00100-01010-00111-00110-01111-10110-11100-11011-11101-10010-01001-01011-00101-10101-10100-00011-00000- -> 370
-00000-00001-01000-11001-10011-11000-10010-10001-11100-00111-00011-01011-01101-10100-10110-00100-11101-10101-11011-01111-10000-01100-01010-00101-01110-11010-10111-01001-00110-00010-00000- -> 435
-00000-00100-10011-11010-01010-11101-00010-11000-01001-10101-11011-01101-00011-10000-01100-10010-01000-00110-10110-11100-10100-01011-11001-01110-01111-10111-00111-00001-00101-10001-00000- -> 397
-00000-01010-00111-11010-00010-10100-10010-10110-11101-00001-01110-01011-11011-11100-01000-11000-10101-10001-01001-10111-00100-10000-00011-00110-01111-00101-01100-11001-10011-01101-00000- -> 395
-00000-00010-00100-01011-10001-01000-00101-10100-11000-11101-00011-01111-11011-10111-01101-01010-10110-10010-00001-01001-10000-11100-01100-01110-11010-00111-00110-11001-10101-10011-00000- -> 485
-00000-11101-01101-00100-10101-00001-01000-01100-00110-01111-01010-10011-00011-10111-11000-00111-10110-11010-11001-10000-01110-01001-00101-10001-01011-10010-11011-00010-10100-11100-00000- -> 436
-00000-10111-01101-10100-01110-00101-00001-01000-11010-10001-10010-01011-00010-10011-11011-10101-01001-11100-01111-11000-10000-11101-00011-00110-01010-11001-00111-10110-01100-00100-00000- -> 393
-00000-11010-00101-01011-10001-10100-10010-00111-10000-11000-00010-00011-11101-00001-01000-11001-11100-00110-01110-01001-00100-11011-01101-10011-01100-01010-10101-00110-01111-11011-00000- -> 431
-00000-01110-11001-00010-10111-10010-11010-01011-00111-00011-01001-10001-01000-11011-00110-01100-11101-00100-01101-11100-10011-10101-01111-00001-01010-11000-10100-00101-10000-10110-00000- -> 431
-00000-11100-10000-01110-00010-01010-01111-10111-00100-10110-00001-11000-00111-10011-10010-01000-11001-11101-10101-10001-10100-00011-00110-01100-11011-01101-11010-01011-00101-01001-00000- -> 381
The best path that was found is:
-00000-11100-10000-01110-00010-01010-01111-10111-00100-10110-00001-11000-00111-10011-10010-11010-00101-01011-01001-01100-00011-11011-10001-10101-00110-01000-01101-10100-11101-11001-00000-
With cost: 330
```