Πολυπλοκότητα

Χωρική Πολυπλοκότητα (Space Complexity)

Κίνητρο

Οι κλάσεις πολυπλοκότητας αντιστοιχούν σε φράγματα σε πόρους.

Ένας τέτοιος πόρος είναι ο *χώρος*: το μέγεθος της μνήμης ενός Η/Υ που χρησιμοποιείται για την επίλυση ενός προβλήματος

Στην περίπτωση ανταιτιοκρατικού Η/Υ, μετράμε το χώρο κάθε κλάδου υπολογισμού και παίρνουμε το μέγιστο.

Κλάσεις Χωρικής Πολυπλοκότητας

Για κάθε συνάρτηση $f:N \rightarrow N$, ορίζουμε:

SPACE($f(n)$)={ L : η L λύνεται από έναν αιτιοκρατικό Η/Υ σε $O(f(n))$ χώρο}

NSPACE($f(n)$)={ L : η L λύνεται από έναν ανταιτιοκρατικό Η/Υ σε $O(f(n))$ χώρο}

PSPACE και NPSPACE

Ορισμός: *PSPACE* είναι το σύνολο των προβλημάτων που είναι επιλύσιμα σε πολυωνυμικό χώρο από μία ΤΜ.

$$PSPACE = \bigcup_c SPACE\left(n^c\right)$$

Αντίστοιχα ορίζεται η NPSPACE:

$$NPSPACE = \bigcup_c NSPACE\left(n^c\right)$$

Χώρος 3-SAT;;;;;

Στιγμιότυπο $_3$SAT με n μεταβλητές και m εκφράσεις.

Δυαδικός μετρητής: Μετράμε από το 0 έως το 2^n-1 δυαδικά.

Αλγόριθμος: Χρησιμοποίησε τον μετρητή.

Ποια είναι η σχέση μεταξύ NP και PSPACE;

3-SAT $\in$ PSPACE

Απόδειξη:

- Δημιουργούμε όλους τους 2^n πιθανούς συνδυασμούς τιμοδοσιών με τον μετρητή.
- Έλεγχος κάθε τιμοδοσίας αν ικανοποιεί όλες τις φράσεις.

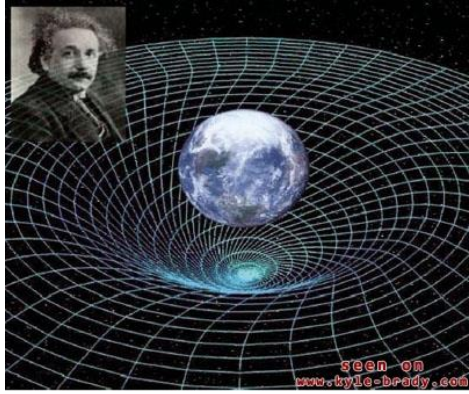
NP $\subseteq$ PSPACE

Απόδειξη:

Έστω πρόβλημα $Y \in$ NP.

Αφού $Y \leq_P$ $_3$SAT, υπάρχει αλγόριθμος που επιλύει το Y σε πολυωνυμικό χρόνο συν ένα πολυωνυμικό πλήθος κλήσεων στο $_3$SAT.

Χωροχρόνος



Έστω ότι ένας τερματίσιμος ανταιτιοκρατικός υπολογισμός χρησιμοποιεί χώρο S

Πόσα βήματα υπολογισμού μπορεί να κάνει;

Αυτός ο αριθμός είναι: $2^{O(S)}$

Το Θεώρημα Savitch

$\forall s(n) \geq n$

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s^2(n))$$

Ιδέα:

Έστω N ένας ανταιτιοκρατικός Η/Υ. με πολ. χώρου $s(n)$.

Κατασκευάζουμε Η/Υ M που ελέγχει όλους τους δυνατούς υπολογισμούς του N.

Αφού κάθε υπολογισμός του N χρησιμοποιεί το πολύ $s(n)$ χώρο, τότε ο M χρησιμοποιεί χώρο το πολύ $s(n)$...

Για την ακρίβεια $2^{O(s(n))}$

Συνέπεια

PSPACE = NPSPACE

Απόδειξη:

Προφανώς, PSPACE$\subseteq$NPSPACE.

Από το θεώρημα του Savitch: NPSPACE$\subseteq$PSPACE.

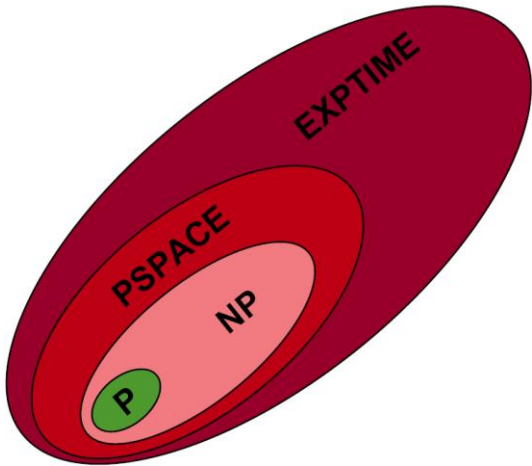
Ιεραρχία

$$EXPTIME = \bigcup_c TIME\left(2^{n^c}\right)$$

$P \subseteq NP \subseteq PSPACE=NSPACE \subseteq EXPTIME$

- Το μόνο που ξέρουμε αυστηρά είναι:

$P \neq EXPTIME$



Χώρος vs Χρόνος

- Η πολυπλοκότητα χώρου δεν συμπεριφέρεται με τον ίδιο τρόπο όπως η πολυπλοκότητα χρόνου:
  - Ο ανταιτιοκρατικός υπολογισμός δεν αυξάνει τη πολυπλοκότητα χώρου δραστικά (Θεώρημα Savitch).

PSPACE-Πληρότητα

- PSPACE: Προβλήματα απόφασης που επιλύονται σε πολυωνυμικό χώρο.

- PSPACE-Πληρότητα: Το πρόβλημα Y είναι PSPACE-πλήρες αν
  1. Το Y είναι PSPACE
  2. Για κάθε πρόβλημα X στο PSPACE, $X \leq_P Y$

PSPACE-δυσχερής: Ισχύει το (2) μόνο

Ποσοδείκτες

- Ένας ποσοδείκτης είναι ένας τελεστής που περιορίζει τις μεταβλητές ενός λογικού τύπου

- Δύο τύποι:
  - Καθολικός $\forall$ (σημαίνει «για κάθε»)
  - Υπαρξιακός $\exists$ (σημαίνει «για κάποιο»)

- Τομέας αναφοράς
  - Σύνολο από το οποίο αντλούν τιμές οι μεταβλητές

Καθολικός Ποσοδείκτης

- Δοθείσης μία πρότασης $P(x)$
- Και τιμών από ένα σύμπαν $x_1 \ldots x_n$
- Το $\forall x\, P(x)$ σημαίνει:

$$P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n)$$

Υπαρξιακός Ποσοδείκτης

- Δοθείσης μίας πρότασης $P(x)$
- Και τιμές από ένα σύμπαν $x_1 \ldots x_n$
- Το $\exists x\, P(x)$ σημαίνει:

$$P(x_1) \vee P(x_2) \vee \ldots \vee P(x_n)$$

Ποσοδείκτες

$\forall x \exists y[y>x]$

$\exists y \forall x[y>x]$

- Εμβέλεια ποσοδείκτη (το τμήμα της πρότασης σε παρενθέσεις στο οποίο αντιστοιχεί ο ποσοδείκτης)
- Όλοι οι ποσοδείκτες στην αρχή:
  - ***Προτακτική Κανονική Μορφή***
- ***Πλήρως ποσοδεδειγμένος τύπος***
  - Κάθε μεταβλητή είναι δέσμια ενός ποσοδείκτη
  - Είναι πάντα Αληθής ή Ψευδής

Το Πρόβλημα QSAT

Quantified SAT

- QSAT: Έστω $\Phi(x_1, \ldots, x_n)$ ένας λογικός τύπος σε μορφή CNF. Είναι ο παρακάτω τύπος ΑΛΗΘΗΣ;

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \forall x_{n-1} \exists x_n \Phi(x_1, \ldots, x_n)$$

- Διαίσθηση: Η Αλίκη επιλέγει 1 για x_1, έπειτα ο Γιώργος για το x_2, έπειτα η Αλίκη για το x_3, κοκ. Μπορεί η Αλίκη να ικανοποιήσει τον Φ για όλες τις επιλογές του Γιώργου;

Παράδειγμα: $\exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

- ΝΑΙ. Η Αλίκη θέτει σε 1 το x_1; Ο Γιώργος θέτει το x_2; Η Αλίκη θέτει το x_3 στην ίδια τιμή με το x_2.

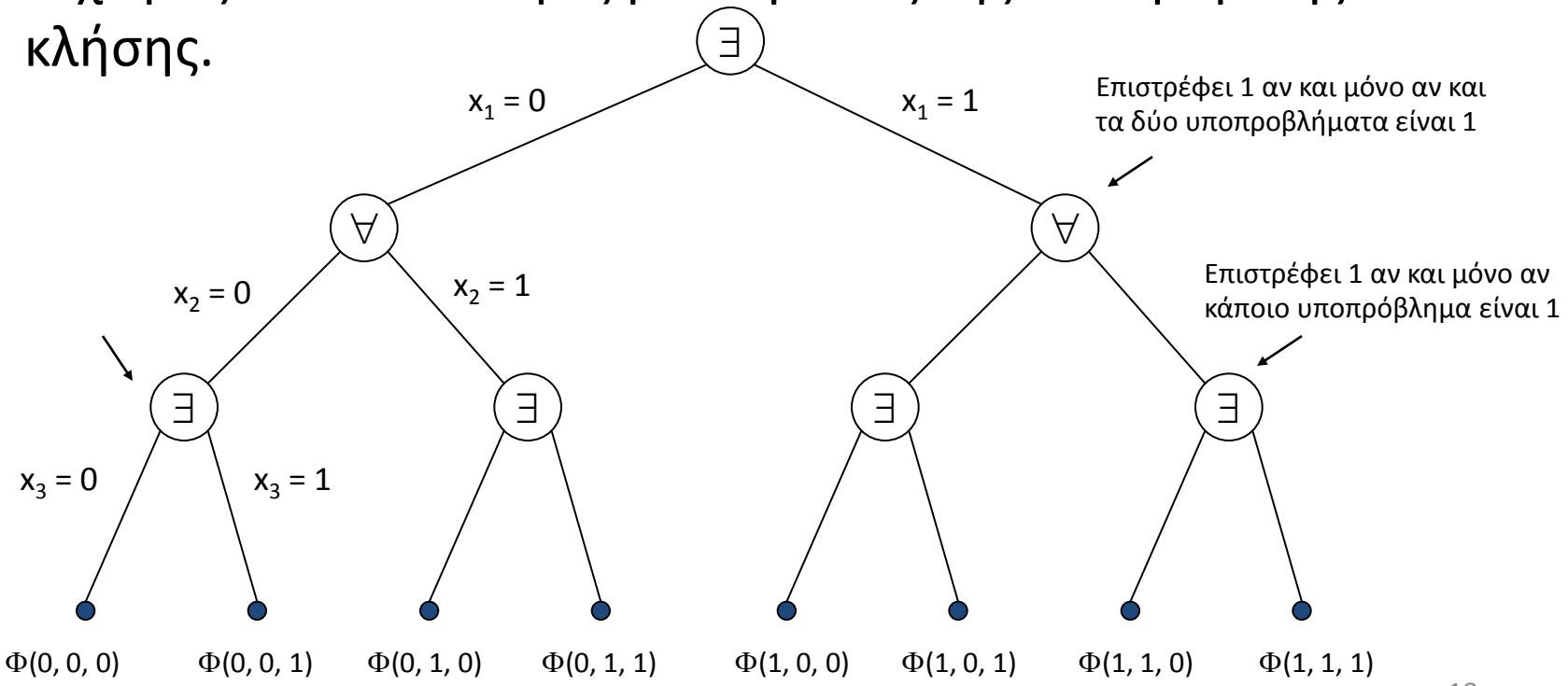
Παράδειγμα: $\exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

- ΟΧΙ. Αν η Αλίκη θέσει 0 στο x_1; Ο Γιώργος θέτει 0 στο x_2 ; Η Αλίκη χάνει
- Αν η Αλίκη θέσει 1 στο x_1; Ο Γιώργος θέτει 1 στο x_2 ; Η Αλίκη χάνει

QSAT $\in$ PSPACE

Απόδειξη: Αναδρομικά δοκιμάζουμε όλους τους συνδυασμούς. Χρειαζόμαστε μόνο ένα bit ανά υποπρόβλημα.

– Ο χώρος είναι ανάλογος με το βάθος της αναδρομικής κλήσης.



QSAT $\in$ PSPACE

```
If the first quantifier is ∃ x_i then
  Set x_i = 0 and recursively evaluate the quantified expression
                    over the remaining variables
  Save the result (0 or 1) and delete all other intermediate work
  Set x_i = 1 and recursively evaluate the quantified expression
                    over the remaining variables
  If either outcome yielded an evaluation of 1, then
     return 1
  Else return 0
  Endif
If the first quantifier is ∀ x_i then
  Set x_i = 0 and recursively evaluate the quantified expression
                    over the remaining variables
  Save the result (0 or 1) and delete all other intermediate work
  Set x_i = 1 and recursively evaluate the quantified expression
                    over the remaining variables
  If both outcomes yielded an evaluation of 1, then
     return 1
  Else return 0
  Endif
Endif
```

QSAT

- Το πρόβλημα αυτό είναι χειρότερο ως προς την δυσκολία σε σχέση με τα NP-πλήρη προβλήματα (μάλλον εκτός του NP και του coNP).

- Το πρόβλημα αυτό είναι το πρότυπο πλήρες πρόβλημα για κλάσεις προβλημάτων ψηλά στην ιεραρχία.

- Παράδειγμα (σκάκι): Μπορεί ο άσπρος να κερδίσει με 4 κινήσεις; Δηλαδή: Υπάρχει $1^{η}$ κίνηση για σένα ($\exists\ a$) έτσι ώστε όποια κίνηση και να κάνει ο Κασπάρωφ ($\forall\ b$), θα κάνεις μία κίνηση ($\exists\ c$) έτσι ώστε ότι κίνηση και να κάνει ο Κασπάρωφ ($\forall\ d$), θα κάνουμε ματ στην $5^{η}$ κίνηση σε ένα έγκυρο παιχνίδι ($\Phi(a,b,c,d)$)?

Βέβαια το σκάκι ως πρόβλημα λύνεται σε O(1)!!!

QSAT

Θεώρημα: Το QSAT είναι PSPACE-πλήρης.

Απόδειξη:

Ότι QSAT $\in$ PSPACE έχει αποδειχθεί.

Για να αποδείξουμε ότι κάθε $A \in$ PSPACE, ανάγεται σε πολυωνυμικό χρόνο στο QSAT, χρησιμοποιούμε μία τακτική αντίστοιχη με αυτή της απόδειξης για το SAT με τη διαφορά ότι την συνδυάζουμε με την αναδρομική τεχνική που εφαρμόστηκε στο Θεώρημα του Savitch.

$_3$QSAT

$_3$QSAT είναι σαν το QSAT μόνο που κάθε φράση έχει 3 λεξιγράμματα. Το $_3$QSAT είναι PSPACE-πλήρες

- Ειδική περίπτωση του QSAT, και άρα ανήκει στο PSPACE
- Αναγωγή από το QSAT (παρόμοια με τα αντίστοιχα NP-πλήρη προβλήματα)

Παιγνία Λογικής

Έστω: $\exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

$$\varphi = \exists x_1 \forall x_2 \ldots Q x_n \Phi(x_1, \ldots, x_n)$$

ένας ποσοδεδειγμένος λογικός τύπος σε προτακτική κανονική μορφή (Q είναι $\exists$ ή $\forall$).

Δύο παίκτες *A* και *B*, επιλέγουν μεταβλητές – ο μεν *A* τις καθολικές και ο δε *B* τις υπαρξιακές.

Νικητής είναι ο *B* αν ο φ είναι ΑΛΗΘΗΣ ενώ νικητής είναι ο *A* αν είναι ΨΕΥΔΗΣ.

Αν ο *A* ή ο *B*, έχει τρόπο πάντα να κερδίζει τότε λέμε ότι έχει ***νικηφόρα στρατηγική***.

Παιγνίο Λογικής

Παιγνίο-Λογικής={$\langle\varphi\rangle$ | στο παιγνίο λογικής για τον τύπο φ ο παίκτης *B* έχει νικηφόρα στρατηγική}

Το πρόβλημα Παιγνίο-Λογικής είναι PSPACE-Πλήρες

Απόδειξη:

Είναι PSPACE-πλήρες αφού συμπίπτει με το QSAT. Πράγματι, ένας τέτοιος τύπος είναι ικανοποιήσιμος αν και μόνο αν υπάρχει νικηφόρα στρατηγική για το αντίστοιχο παιγνίο λογικής

Το Παιχνίδι της Γεωγραφίας

Παιχνίδι: Δύο παίκτες, *A* και *B*, που εναλλάσονται μεταξύ τους ονομάζοντας πόλεις. Κανόνες: Η επόμενη πόλη θα πρέπει:

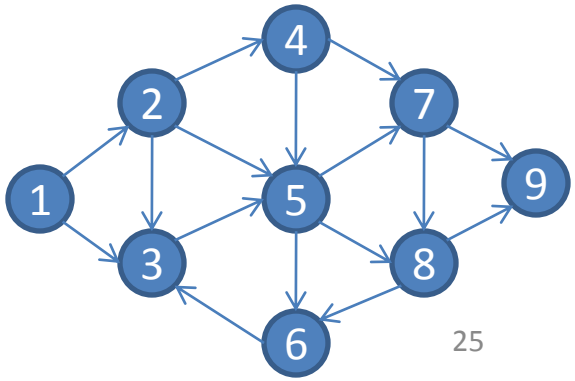
- Να ξεκινά από το τελευταίο γράμμα της προηγούμενης λέξης
- Και να διαφέρει από όλες τις προηγούμενες

Ένας παίκτης κερδίζει αν ο αντίπαλός του δεν μπορεί να αναφέρει καμία πόλη.

Γενίκευση: Είσοδος: Γράφημα *G* του οποίου οι κόμβοι είναι οι πόλεις και οι ακμές αντιστοιχούν στον 1ο κανόνα (ποια πόλη ακολουθεί ποια). Το παιχνίδι ξεκινά από κάποιο κόμβο *s*, με τον παίκτη *A* να ξεκινά.

Ερώτηση: Έχει ο παίκτης *A* νικηφόρα στρατηγική;

Η γενικευμένη γεωγραφία είναι PSPACE-πλήρης



Γεωγραφία $\in$ PSPACE

Αναδρομικός αλγόριθμος win(G,s) για να προσδιορίσουμε αν ο παίκτης που παίζει πρώτος κερδίζει το παιχνίδι ξεκινώντας από το s

- Αν δεν υπάρχει ακμή (s,v) στο G τότε επέστρεψε ΟΧΙ, αλλιώς για κάθε ακμή (s,v) στο G, καλούμε την win αναδρομικά στο (G-{s},v) και αν κάποια από αυτές τις κλήσεις απαντήσει ΟΧΙ τότε επέστρεψε ΝΑΙ αλλιώς ΟΧΙ
- Βάθος αναδρομής: #γύρων = n =# κόμβων του G
- Στοίβα αναδρομής = ακολουθία κόμβων που σβήστηκαν από το γράφημα

$\Rightarrow$ πολυωνυμικός χώρος

${}_3\text{QSAT} \leq_p$ Γεωγραφία

Έστω ότι οι ποσοδείκτες εναλλάσσονται έτσι ώστε να ξεκινάνε και να τελειώνουν με το ∃:

Είσοδος: $\Phi = \exists y_1 \, \forall y_2 \, \exists y_3 \ldots \exists y_m \, \varphi(y_1, y_2, \ldots, y_m)$

όπου $\varphi = c_1 \wedge c_2 \wedge \ldots \wedge c_k$

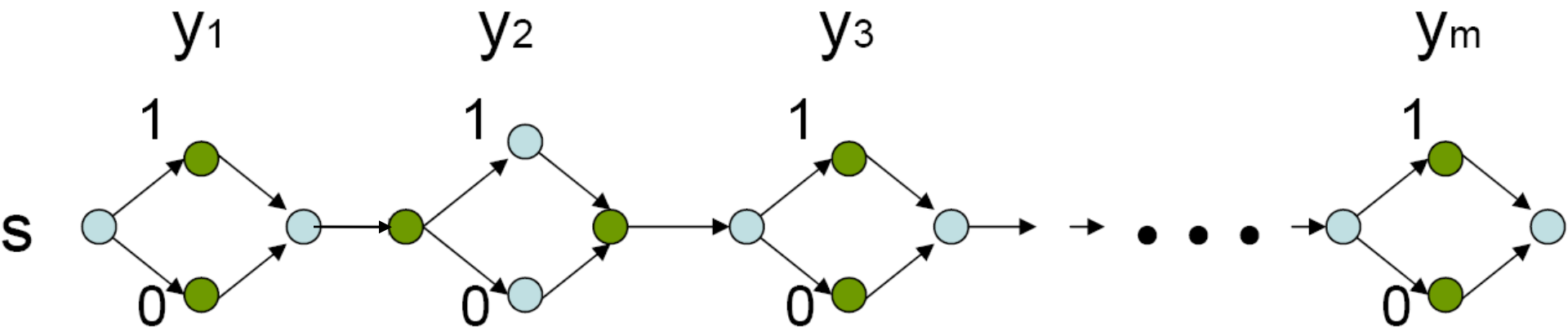
Ο παίκτης *A* (υπαρξιακός) και ο παίκτης *B* (καθολικός).

Κατασκευάζουμε γράφημα *G* με αρχικό κόμβο *s*, έτσι ώστε ο *A* να έχει νικηφόρα στρατηγική αν και μόνο αν η Φ είναι ΑΛΗΘΗΣ.

Το γράφημα έχει:

- Ένα κομμάτι που αντιστοιχεί στις μεταβλητές, όπου οι παίκτες επιλέγουν τιμοδοσία
- Και ένα κομμάτι για τις φράσεις για έλεγχο των τιμοδοσιών

Οι Μεταβλητές

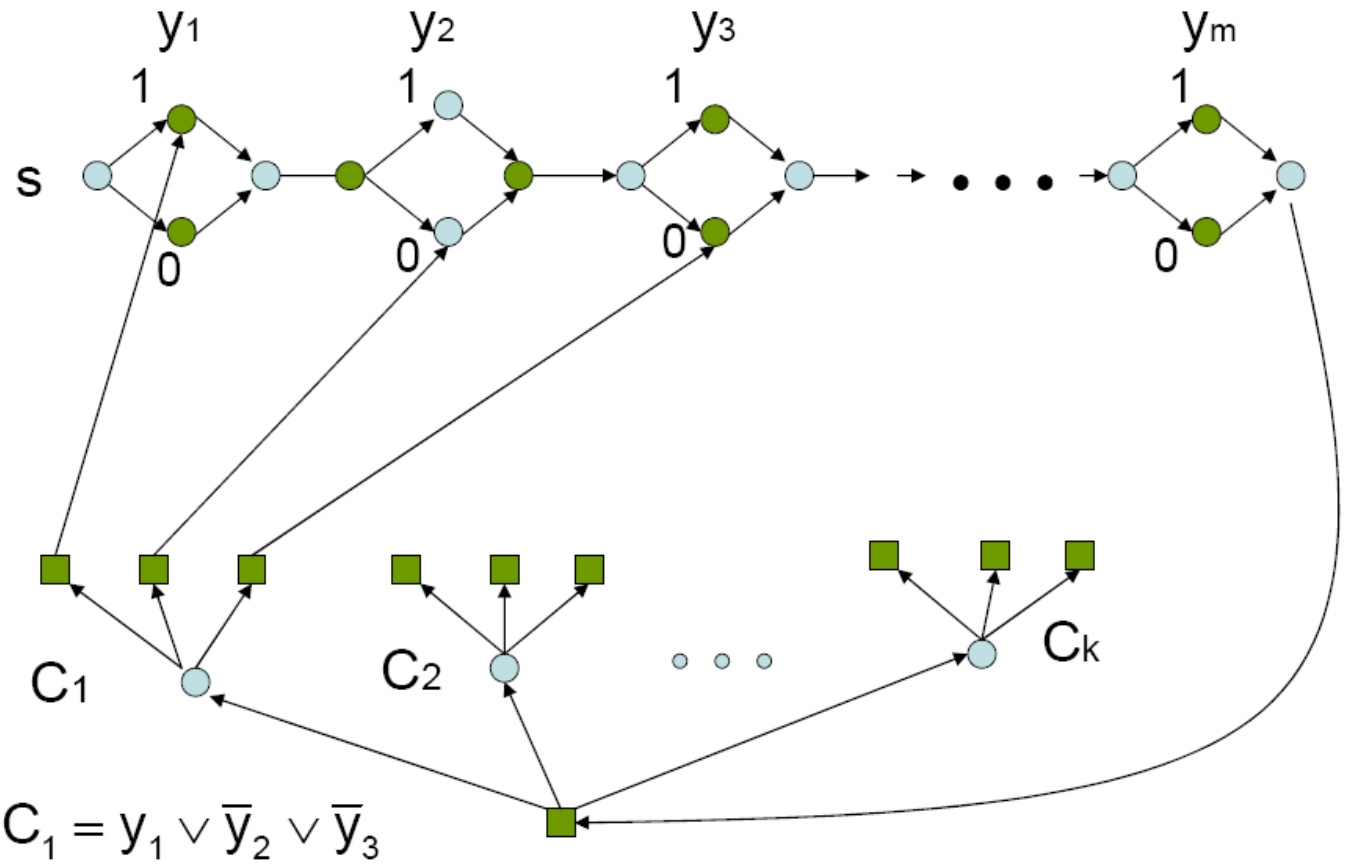


Γαλάζιοι ρόμβοι: Παίκτης *A* επιλέγει μονοπάτι

Πράσινοι ρόμβοι: Παίκτης *B* επιλέγει μονοπάτι

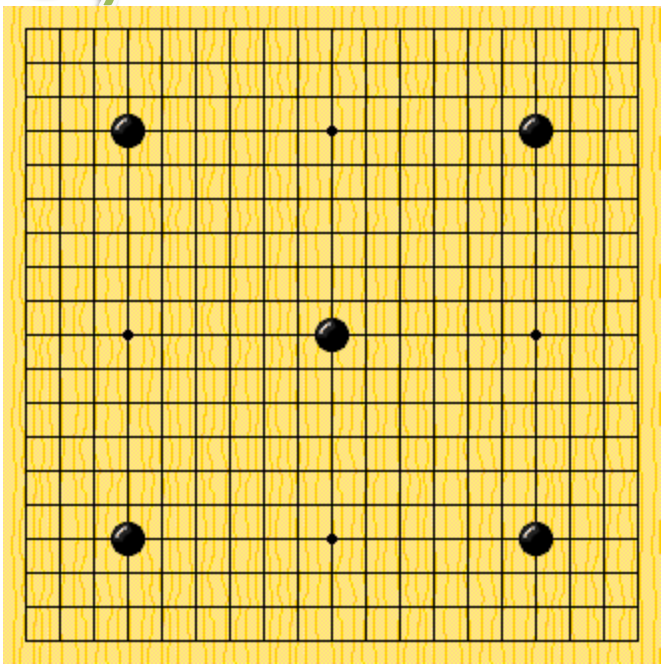
Στο αρχικό μονοπάτι που αφορά τις μεταβλητές οι παίκτες εναλλάσσονται και επιλέγουν τιμοδοσίες για τις μεταβλητές που τους αντιστοιχούν (*A* υπαρξιακές και *B* καθολικές)

Οι Φράσεις

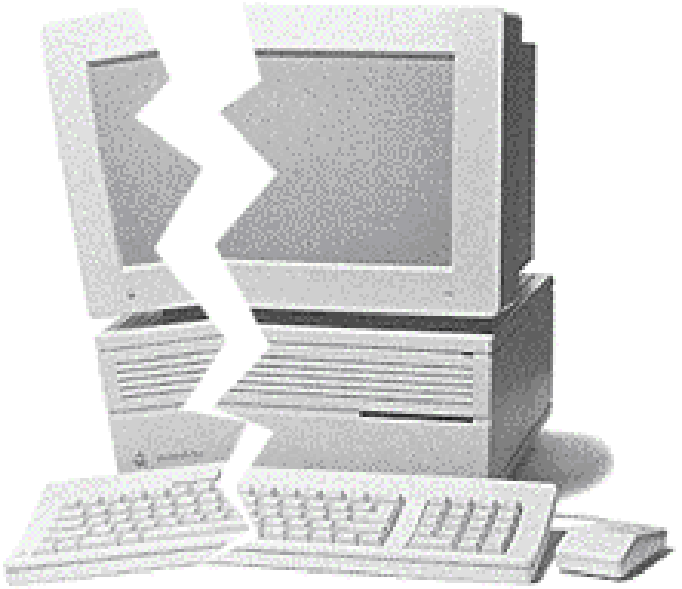


- Αν μία φράση δεν ικανοποιείται, τότε ο παίκτης *B* την επιλέγει και κερδίζει

Το Παιχνίδι GO είναι PSPACE-δυσχερές



Οι παίκτες σε σειρά τοποθετούν πέτρες σε 361 τομές από ένα πλέγμα 19×19. Ο μαύρος παίζει πρώτος. Εννιά σημεία δίνονται εκ των προτέρων για να υπάρχει ισοδυναμία μεταξύ δύο παικτών. Κάθε τομή είναι ένα σημείο ενώ κάθε πέτρα κερδισμένη είναι επίσης ένας πόντος.



Άρα οι υπολογιστές δεν μπορούν να παίξουν!

Το Go είναι τόσο πολύπλοκο που τα καλύτερα προγράμματα χάνουν από παιδιά που παίζουν GO. Οι προγραμματιστές το αποκαλούν «το τελευταίο οχυρό της ανθρώπινης διάνοιας».

PSPACE-πλήρη Προβλήματα

- Ανταγωνιστική Χωροθέτηση Υπηρεσιών

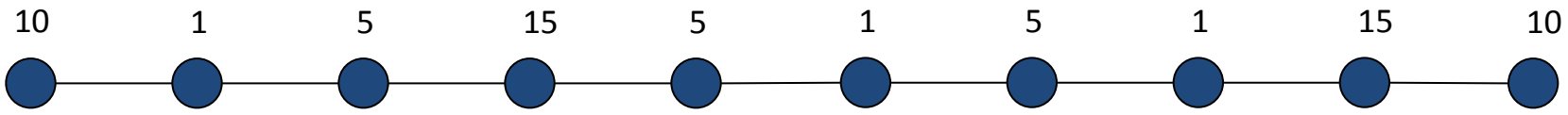
- Φυσικές γενικεύσεις παιχνιδιών.
  - Othello, Hex, Geography, Rush-Hour, Instant Insanity
  - Shanghai, go-moku, Sokoban

- Είναι δυνατό να μετακινήσουμε και να περιστρέψουμε ένα πολύπλοκο αντίκειμενο μέσα από έναν ακανόνιστου σχήματος διάδρομο;

- Είναι δυνατό να υπάρχει αδιέξοδο (deadlock) σε ένα σύστημα επεξεργαστών που επικοινωνούν μεταξύ τους;

Ανταγωνιστική Χωροθέτηση Υπηρεσιών

- Είσοδος: Γράφημα με θετικά βάρη και ένας στόχος *B*.
- Παιχνίδι: Δύο ανταγωνιστικοί παίκτες εναλλάσσονται στην επιλογή κόμβων. Δεν επιτρέπεται να επιλέξουν κόμβο αν κάποιος γείτονας έχει ήδη επιλεγεί.
- Μπορεί ο 2ος παίκτης να εγγυηθεί ένα κέρδος τουλάχιστον B;



<u>ΝΑΙ</u> αν B = 20; **<u>ΟΧΙ</u>** αν B = 25.

Ανταγωνιστική Χωροθέτηση Υπηρεσιών

Το πρόβλημα είναι PSPACE-πλήρες.

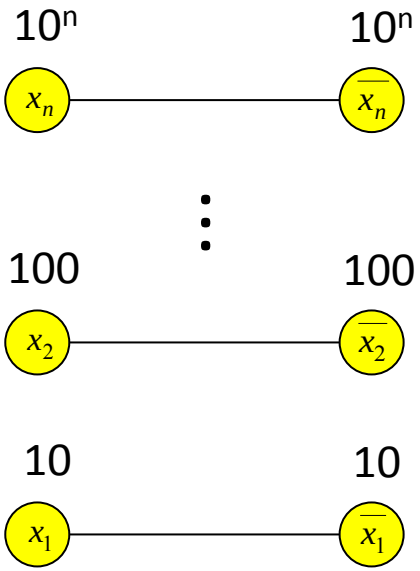
- Απόδειξη:
  - Για επίλυση σε πολυωνυμικό χώρο χρησιμοποιούμε αναδρομή όπως και στο QSAT, αλλά σε κάθε βήμα υπάρχουν μέχρι *n* επιλογές αντί για 2.
  - Για να δείξουμε ότι είναι πλήρες, θα δείξουμε ότι το $_3$QSAT ανάγεται σε αυτό.

Ανταγωνιστική Χωροθέτηση Υπηρεσιών

έστω n περιττός

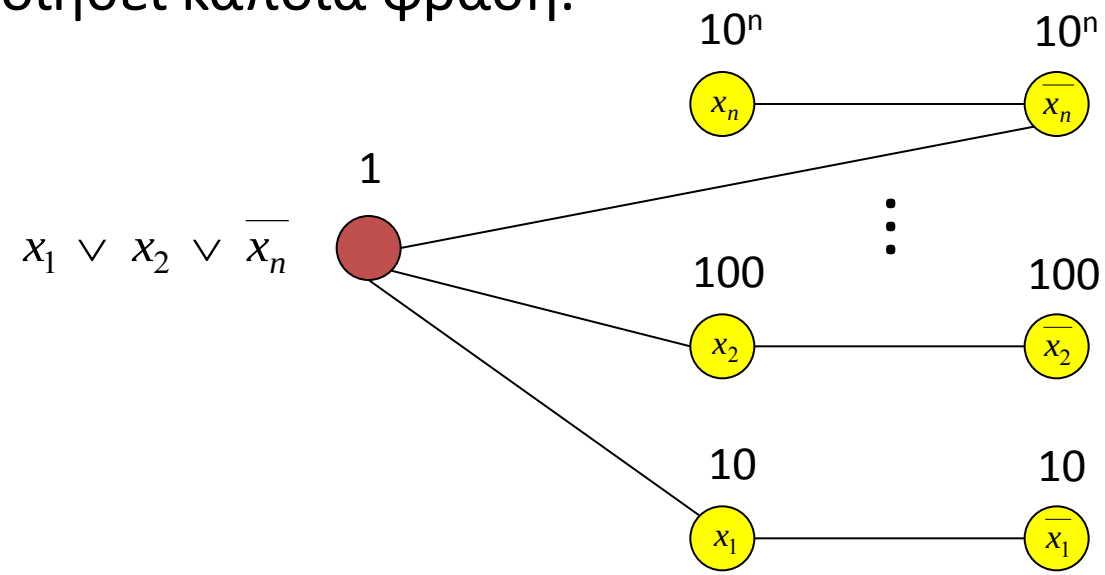
Στιγμιότυπο $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ του ${}_3$QSAT.

- Έχουμε έναν κόμβο για κάθε μεταβλητή και το συμπλήρωμά της και τα συνδέουμε.
  - Το πολύ ένα μεταξύ του x_i και του συμπληρώματός του μπορεί να επιλεχθεί
- Επιλέγουμε $c \geq k+2$, και θέτουμε βάρος c^i στο x_i και το συμπλήρωμά του $B = c^{n-1} + c^{n-3} + \ldots + c^4 + c^2 + 1$.
  - Εξασφαλίζει ότι οι μεταβλητές επιλέγονται σε σειρά $x_n, x_{n-1}, \ldots, x_1$.
- Ως έχει ο παίκτης 2 θα χάσει για 1 μονάδα: $c^{n-1} + c^{n-3} + \ldots + c^4 + c^2$.



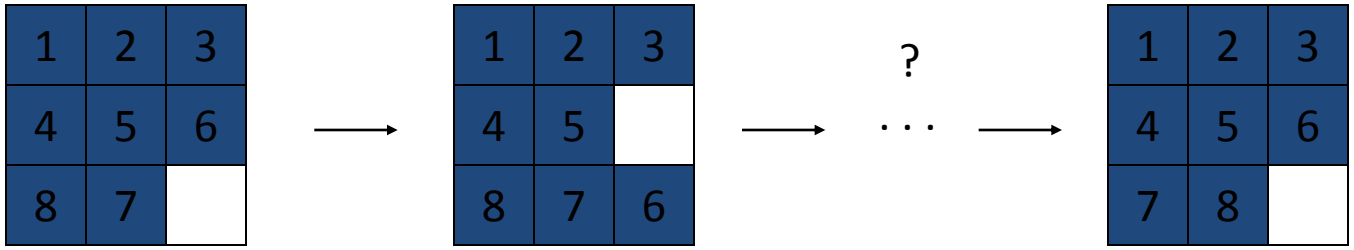
Ανταγωνιστική Χωροθέτηση Υπηρεσιών

- Δώσε στον παίκτη 2 μία τελευταία κίνηση όπου θα προσπαθήσει να κερδίσει.
- Για κάθε φράση C_j, προσθέτουμε ένα κόμβο με τιμή 1 και μία ακμή στα λεξιγράμματά της.
- Ο παίκτης 2 μπορεί να κάνει την τελευταία κίνηση αν και μόνο αν η τιμοδοσία που δόθηκε από τους παίκτες εναλλάξ αποτυγχάνει να ικανοποιήσει κάποια φράση. ▪



15-Γρίφος

- 8-γρίφος, 15-γρίφος
  - Πλαίσιο: 3×3 πλαίσιο από πλακίδια αριθμημένα από 1-8.
  - Κίνηση: μετακίνηση γειτονικού πλακιδίου σε άδειο τετράγωνο (άσπρο)
  - Στόχος: Εύρεση ακολουθίας κινήσεων από αρχική διαμόρφωση σε διαμόρφωση στόχο



Το Πρόβλημα Σχεδιασμού

Συνθήκες: $C = \{ C_1, \ldots, C_n \}$.

Αρχική Διαμόρφωση: $C_0 \subseteq C$ συνθηκών που αρχικά ικανοποιούνται

Διαμόρφωση Στόχος: $C^* \subseteq C$ συνθηκών που θέλουμε να ικανοποιούνται

Τελεστές: $O = \{ O_1, \ldots, O_k \}$.

- Η εφαρμογή του τελεστή O_i, προϋποθέτει κάποιες συνθήκες
- Μετά την εφαρμογή του O_i κάποιες συνθήκες γίνονται Αληθείς και κάποιες Ψευδείς

- Σχεδιασμός: Είναι δυνατό εφαρμόζοντας μία ακολουθία τελεστών να μετακινηθούμε από την αρχική διαμόρφωση στη διαμόρφωση στόχο;

- Παράδειγμα
  - 15-γρίφος.
  - Ο κύβος τους Rubik.
  - Logistics (μετακίνηση ανθρώπων, εξοπλισμού και υλικών)

Ο 8-Γρίφος

- Παράδειγμα Σχεδιασμού. Μπορούμε να λύσουμε τον 8-γρίφο;
- Συνθήκες: C_{ij}, $1 \leq i, j \leq 9$. ← C_{ij} Σημαίνει ότι το πλακίδιο i είναι στο τετράγωνο j
- Αρχική διαμόρφωση: $C_0 = \{C_{11}, C_{22}, \ldots, C_{66}, C_{78}, C_{87}, C_{99}\}$.
- Διαμόρφωση Στόχος: $C^* = \{C_{11}, C_{22}, \ldots, C_{66}, C_{77}, C_{88}, C_{99}\}$.
- Τελεστές.
  - Προϋπόθεση για εφαρμογή O_i = $\{C_{11}, C_{22}, \ldots, C_{66}, C_{78}, C_{87}, C_{99}\}$.
  - Έπειτα από τον O_i, οι συνθήκες C_{79} και C_{97} είναι αληθείς.
  - Έπειτα από τον O_i, οι συνθήκες C_{78} και C_{99} είναι ψευδείς.
- Λύση: Δεν υπάρχει λύση πάντα για τον 8-γρίφο ή τον 15-γρίφο!

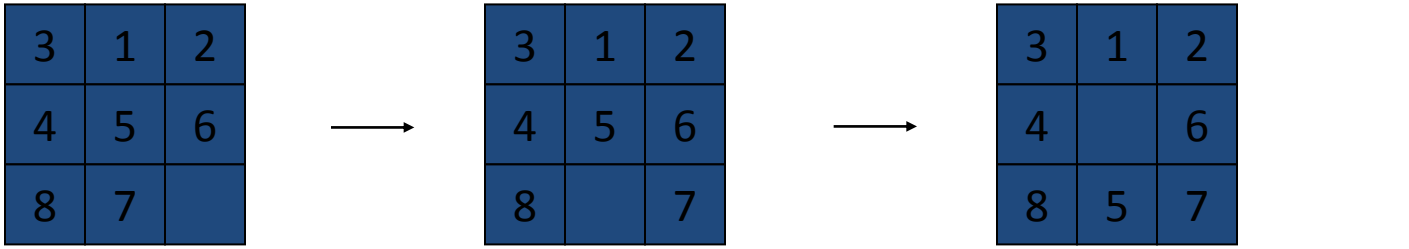
1	2	3
4	5	6
8	7	9

O_i

1	2	3
4	5	6
8	9	7

Γιατί ο 8-Γρίφος δεν είναι Επιλύσιμος;

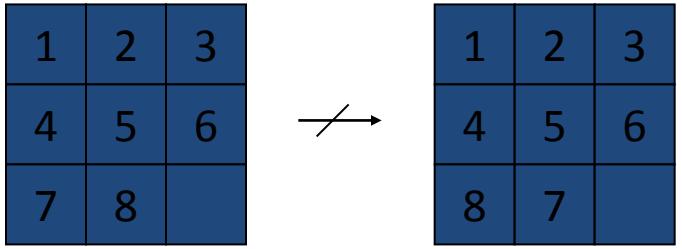
- Αναλλοίωτη Ιδιότητα του 8-γρίφου: Κάθε κίνηση (νόμιμη) διατηρεί την ισοτιμία του πλήθους των ζευγών πλακιδίων σε αντίστροφη σειρά (αναστροφές).



3 αναστροφές
1-3, 2-3, 7-8

3 αναστροφές
1-3, 2-3, 7-8

5 αναστροφές
1-3, 2-3, 7-8, 5-8, 5-6



0 αναστροφές

1 αναστροφή: 7-8

Πρόβλημα Σχεδιασμού: Δυαδικός Μετρητής

- Μπορούμε να αυξήσουμε μοναδιαία έναν μετρητή με n-bit από όλα 0 σε όλα 1;

- Συνθήκες: $C_1, \ldots, C_n$. ← Το C_i αντιστοιχεί στο bit i = 1
- Αρχική διαμόρφωση: $C_0 = \phi$.
- Τελική διαμόρφωση: $C^* = \{C_1, \ldots, C_n\}$.
- Τελεστές: $O_1, \ldots, O_n$.
  - Ο τελεστής O_i εφαρμόζεται όταν ισχύουν τα $C_1, \ldots, C_{i-1}$.
  - Μετά τον O_i, η συνθήκη C_i γίνεται αληθής.
  - Μετά τον O_i, οι συνθήκες $C_1, \ldots, C_{i-1}$ γίνονται ψευδείς.

- Λύση: $\{\} \Rightarrow \{C_1\} \Rightarrow \{C_2\} \Rightarrow \{C_1, C_2\} \Rightarrow \{C_3\} \Rightarrow \{C_3, C_1\} \Rightarrow \ldots$

- Κάθε λύση απαιτεί τουλάχιστον $2^n - 1$ βήματα.

Πρόβλημα Σχεδιασμού: Εκθετικός Χώρος και Χρόνος

- Γράφημα Διαμόρφωσης G.
  - Κόμβος για κάθε μία από τις 2^n πιθανές διαμορφώσεις.
  - Μία ακμή από διαμόρφωση c' στη διαμόρφωση c'' αν κάποιος από τους τελεστές μετατρέπει την c' σε c''.

- Υπάρχει διαδρομή από το C_0 στο C* στο γράφημα διαμόρφωσης;

Ο Σχεδιασμός ανήκει στο EXPTIME.

<u>Απόδειξη:</u>

BFS για εύρεση διαδρομής από C_0 σε C* στο γράφημα.

Το γράφημα μπορεί να έχει 2^n κόμβους και η συντομότερη διαδρομή μπορεί να είναι μήκους 2^n - 1.

Σχεδιασμός ∈ PSPACE

Απόδειξη:

- Έστω διαδρομή από c_1 σε c_2 μήκους L.
- Μία διαδρομή από c_1 στον μέσο κόμβο και μετά στο c_2 έχει μήκος ≤ L/2.
- Κοιτάμε όλους τους πιθανούς μέσους κόμβους.
- Εφαρμόζουμε αναδρομικά. Βάθος αναδρομής = $\log_2$ L. ▪

```
boolean hasPath(c1, c2, L) {
   if (L ≤ 1) return correct answer

                                  Χρηαιμοποίησε δυαδικό μετρητή

   foreach configuration c' {
      boolean x = hasPath(c1, c', L/2)
      boolean y = hasPath(c2, c', L/2)
      if (x and y) return true
   }
   return false
}
```